Coding Theory and Applications

Linear Codes

Enes Pasalic
University of Primorska
Koper, 2013

Contents

Chapter 1

Preface

This book has been written as lecture notes for students who need a grasp of the basic principles of linear codes.

The scope and level of the lecture notes are considered suitable for undergraduate students of Mathematical Sciences at the Faculty of Mathematics, Natural Sciences and Information Technologies at the University of Primorska.

It is not possible to cover here in detail every aspect of linear codes, but I hope to provide the reader with an insight into the essence of the linear codes.

Enes Pasalic
enes.pasalic@upr.si

Chapter 2

Shannon theory and coding

Contents of the chapter:

- Mariners
- Course description
- Decoding problem
- Hamming distance
- Error correction
- Shannon

Coding theory - introduction

Coding theory is fun (to certain extent :)

Can we live without error correction codes ?

– **Probably not** !!

What would you miss :

You would not be able to listen CD-s, retrieve correct data from your hard disk, would not be able to have a quality communication over telephone etc.

Communication, storage errors, authenticity of ISBN numbers and much more is protected by means of error-correcting codes.

Students' favorite application

- One of the most popular applications in CD players
- CD records becomes scratchy (the quality is getting worse)
- Each tiny scratch would cause a noise when listening the music (worse than vinyl)
- Problem: Invent a good code that can correct burst errors (consecutive errors)
- Solution: Use an encoder and decoder based on the Reed-Solomon codes !

Coding theory - repetition code

- Most of the storage media is prone to errors (CDs, DVDs, magnetic tapes).
- In certain applications errors in retrieved data are not acceptable.
- Need some redundancy of information, i.e. instead of saving 1 and 0 we can save 000 and 111.
- Example of a simple repetition code
- How do we retrieve the information - simply if no error $000 \to 0$ and $111 \to 1$.
- If only one error then majority rules,

$$000, 001, 010, 100 \to 0$$
$$111, 101, 110, 011 \to 1$$

Coding theory - repetition code II

- What about correcting 2 errors ? Nothing we can do with this code, e.g. $000 \to 110$ and we decode 0 as 1 !
- Why not use repetition code of length 5 ? Then we can correct up to 2 errors ?
- Indeed $00000 \to 00011$ it is still decoded as 0 !
- The problem is that this approach is not quite efficient 5 times more data.
- One of the main goals of coding theory is to increase efficiency.
- Main idea is to encode a block of bits and not a single bit !

Coding efficiency

- For instance Hamming code takes a block of $k = 4$ bits and encode it into a block of $n = 7$ bits; still can correct 1 error !

  Comparison:

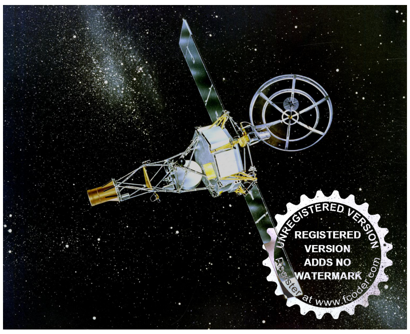
  - Repetition code: 1 bit encoded as 3 bits
  - Hamming code: 4 bits encoded as 7 bits

- We may talk about coding efficiency (code rate) - clearly the Hamming code is better; using less redundancy for the same error correction capability.

- We may wish to correct more than a few errors in a codeword - other codes such as Reed-Muller code exist.

Mariner story

- Back in 1969 Mariners (Voyagers etc.) were supposed to send pictures from Mars to Earth
- The problem was a thermal noise to send pixels with grey scale of 64 level.



- Redundancy was introduced - 6 bits (64 scale grades) encoded as a 32-bit tuple.

Mariner story - encoding

- Such an encoding could correct up to 7 errors in transmission.
- Correcting errors is **not for free**- we have to send bits 32/6 times faster.
- This means that the total energy per bit is reduced - this causes increased probability of (bit) error !
- Have we overpaid the capability of correcting errors ?
- The answer lies in computing coding gain - if positive then we save energy (reduce the probability of error).

Error probability in noisy channels

- Assume that a transmitter has a total energy per bit E_b available. E.g. to send "1" a signal with amplitude $s = \sqrt{E_b}$ is sent and $s = -\sqrt{E_b}$ for "0".
- In presence of AWGN (Additive White Gaussian Noise) the received signal is
$$r = s + n,$$
n has zero mean and variance σ^2.
- Hard decision decoding: $r > 0$ "1" sent; "0" otherwise. Then the bit error probability is,
$$p_e = \int_{\sqrt{E_b}}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-y^2}{2\sigma^2})dy = Q\left(\sqrt{\frac{E_b}{\sigma^2}}\right).$$

Error probability for Mariner

- Assumption: Each block of 6 bits may be wrong with probability $P_E < 10^{-4}$.
- In case of no coding we need $E_b/\sigma^2 = 17.22$ as,

$$p_e = Q(\sqrt{17.22}) \approx 10^{-4}/6 \text{ and } P_E = 1 - (1 - p_e)^6 \approx 10^{-4}.$$

- Compute p_e for given P_E and get SNR=E_b/σ^2.
- In Mariner 6 bits encoded as 32 bits, i.e. energy per bits decreases:

$$p'_e = Q\left(\sqrt{\frac{6E_b}{32\sigma^2}}\right)$$

- For given $SNR = 17.22$ $p'_e = 0.036$ – 2000 times larger than p_e

Coding gain for Mariner

- The benefit is in error correction. After decoding 32 bits to 6 bits,

$$P'_E = \sum_{i>7} \binom{32}{i} (p'_e)^i (1 - p'_e)^{32-i} \approx 1.4 \cdot 10^{-5}.$$

- Even better results if *soft decoding* is used.
- The use of coding may be viewed as saving the energy ! The code used in Mariner was a $[32, 6]$ Reed-Muller code.
- For Mariner example to get $P'_E = 10^{-4}$ an SNR of 14.83 is required (instead of 17.22).

Definition The ratio between SNR (uncoded) and SNR (coded) for equal error probability after decoding is called the coding gain.

ISBN

The International Standard Book Number (ISBN) is a 10-digit codeword such as 0-521-55374-1.

- The first digit indicates the language (0 or 1 for English).
- The next group of digits specifies the publisher (521 for Cambridge University Press).
- The next group of 5 digits forms the book number assigned by the publisher (the groups of digits are of variable length).
- The final digit x_{10} is chosen so that the entire number $x_1 x_2 \dots x_{10}$ satisfies the following check equation:

$$\sum_{i=1}^{10} x_i = 0 \pmod{11}.$$

ISBN - example

The redundant bit offers a simple error correction.

Example The sixth digit in the ISBN $0 - 7923 - \square 519 - X$ has faded out. We want to find the missing digit.

– When $x_{10} = 10$ the value is represented by the letter X.

The missing digit x_6 satises the equation, modulo 11,

$0 = 1 \cdot 0 + 2 \cdot 7 + 3 \cdot 9 + 4 \cdot 2 + 5 \cdot 3 + 6 \cdot x_6 + 7 \cdot 5 + 8 \cdot 1 + 9 \cdot 9 + 10 \cdot 10,$

which gives $6x_6 = 9 \pmod{11}$, i.e. $x_6 = 7$.

Course topics

- The following topics will be covered in the course:

1. Linear codes with emphasis on Hadamard codes
2. Golay and Reed-Muller codes
3. Cyclic codes and BCH codes
4. Reed-Solomon codes and perfect codes
5. Constructing new codes from known ones
6. Asymptotically good codes and algebraic geometry codes
7. Bounds on codes and convolutional codes ...

Block code

Definition A *block code* of length n containing M codewords over the alphabet A is a set of M n-tuples where each n-tuple takes its components from A. Denoted $[n, M]$ code over A.

Example Let $A = \{0, 1\}$ and consider a $[5, 4]$ code defined by its codewords:

$$\begin{array}{ll} c_0 = (00000) & c_1 = (10110) \\ c_2 = (01011) & c_3 = (11101) \end{array}$$

- What are the properties of such a code ? Linearity, rate, error-correcting capability etc.
- Linearity is (almost) obvious $c_1 + c_2 = c_3$ using bitwise modulo two addition !

Redundancy of the code

- How many information bits can be carried over ?
- Total of 4 codewords means that 2 bits are transmitted.
- Redundancy measures amount of extra bits

$$r = n - k$$

In our case $n - k = 5 - 2 = 3$. Three extra bits for the purpose of correcting/detecting errors !
- Need to specify the mapping from information to codewords.
- E.g. we may have,

$$(00) \mapsto (00000) \quad (01) \mapsto (01011)$$
$$(10) \mapsto (10110) \quad (11) \mapsto (11101)$$

Rate of the code

Definition The *rate* of an $[n, M]$ code which encodes information k-tuples is

$$R = \frac{k}{n} = \frac{\log_{|A|} M}{n}.$$

- In our example the rate is $R = \frac{2}{5}$, good or bad ?

– Hard to answer - several issues to be considered :

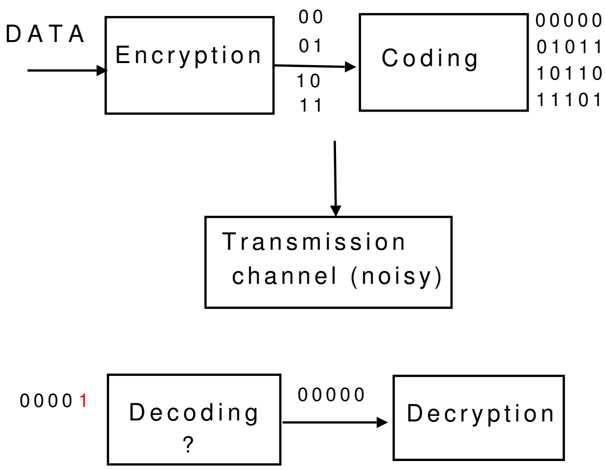
- Depends on application; how many errors we need to correct and what is the error probability of the channel
- What we do know: There exist codes of long length $(n \to \infty)$ so that the probability of error after decoding $\to 0$!!

Coding Alphabet

- In the previous example we assumed the alphabet $A = \{0, 1\}$.
- Easiest case - binary. We consider in general:
  - A is q-ary alphabet
  - $q = 2$, $q = p > 2$, $q = p^m$ or sometimes
  - $A = \{a, b, c, d\}$

In general, increasing the coding alphabet may improve the performance of the code, but decoding complexity is a problem

Transmission scheme



Decoding problem

– Given an $[n, M]$ code C received vector $\mathbf{r}$ there are several choices:

- no errors have occurred - accept $\mathbf{r}$ as a sent codeword
- errors have occurred; correct $\mathbf{r}$ to a codeword $\mathbf{c}$
- errors have occurred - no correction possible

Three main strategies (depends on the application):

1. error correction
2. error detection (retransmission request)
3. Hybrid approach both correction and detection

Hamming distance - definition

Definition The *Hamming distance* $d(\mathbf{x},\mathbf{y})$ between two codewords $\mathbf{x}$ and $\mathbf{y}$ is the number of coordinate positions in which they differ.

- E.g. the Hamming distance between $\mathbf{x}=$ 01011 and $\mathbf{y}$=10110 is 4.

The Hamming distance of an $[n, M]$ code is a minimum distance between any two codewords

$$d = \min_{\mathbf{x},\mathbf{y}\in\mathbf{C}} d(\mathbf{x},\mathbf{y}).$$

- Computing the minimum distance of the code requires calculating $\binom{M}{2} \approx \frac{M^2}{2}$ Hamming distances.

Hamming distance - properties

Three simple properties:

1. $d(\mathbf{x}, \mathbf{y}) \geq 0$
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
3. $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$ - triangle inequality (exercise)

- Nearest neighbor decoding (minimum distance) uses the Hamming distance in decoding.

IDEA: Given a received n-tuple r find the closest codeword c to r (if it exists) and correct r to c

- What if several codewords are equally close ?
- Either retransmission or pick up a codeword at random.

Maximum likelihood decoding

- Nearest neighbor decoding justified through *maximum likelihood decoding*.
- IDEA: Maximize the probability

$$max_{\mathbf{c} \in \mathbf{C}} Pb(\mathbf{r}, \mathbf{c}),$$

$Pb(\mathbf{r}, \mathbf{c})$ - the probability that $\mathbf{r}$ is received, given that $\mathbf{c}$ is sent.

- Assumptions:
  - A code with an alphabet of q symbols
  - p error probability for each symbol
- If $d(\mathbf{r}, \mathbf{c}) = \mathbf{d}$ then

$$Pb(\mathbf{r}, \mathbf{c}) = (1-p)^{n-d}\left(\frac{p}{q-1}\right)^{d}.$$

Maximum likelihood decoding II

- Suppose $\mathbf{c}_1$ and $\mathbf{c}_2$ are two codewords, and $\mathbf{r}$ is received. Furthermore assume $d(\mathbf{r}, \mathbf{c}_1) = d_1 \leq d(\mathbf{r}, \mathbf{c}_2) = d_2$.
- Wonder when $Pb(\mathbf{r}, \mathbf{c}_1) \geq Pb(\mathbf{r}, \mathbf{c}_2)$?
- If this holds then

$$(1-p)^{n-d_1}(\frac{p}{q-1})^{d_1} > (1-p)^{n-d_2}(\frac{p}{q-1})^{d_2}$$

so that

$$(1-p)^{d_2-d_1} > (\frac{p}{q-1})^{d_2-d_1} \Rightarrow \left(\frac{p}{(1-p)(q-1)}\right)^{d_2-d_1} < 1$$

- Thus, $d_2 \geq d_1$ implies for $p < \frac{q-1}{q}$ the max. likelihood is sound.

Decoding using the maximum likelihood - example

- Again let $\mathbf{C} = \{00000, 10110, 01011, 11101\}$ and $p = 0.1$. If $\mathbf{r} = (11111)$ is received then,

$$\begin{aligned}
Pb(\mathbf{r}, 00000) &= (0.1)^5 = 0.00001 \\
Pb(\mathbf{r}, 10110) &= (0.1)^2(0.9)^3 = 0.00729 \\
Pb(\mathbf{r}, 01011) &= (0.1)^2(0.9)^3 = 0.00729 \\
Pb(\mathbf{r}, 11101) &= (0.1)^1(0.9)^4 = 0.06561
\end{aligned}$$

- $Pb(\mathbf{r}, 11101)$ is largest, thus $\mathbf{r}$ is decoded as 11101.

One error could be corrected, but we may be satisfied only with detection of errors. How many errors we can detect ?

Error correcting capability

Theorem If C is an $[n, M]$ code with $d \geq 2e + 1$, then C can correct up to e errors. If used for error detection only, C can detect $2e$ errors.

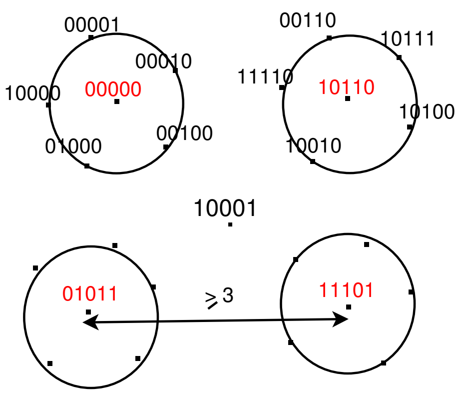
Proof (Sketch) Let $\mathbf{c_i}$, $1 \leq i \leq M$ be the codewords of C and define

$$S_{c_i} = \{\mathbf{x} \in A^n : d(\mathbf{x}, \mathbf{c_i}) \leq e\}$$

where A is alphabet of C and S_{c_i} is sphere of radius e around the $\mathbf{c_i}$. Then $S_{c_i} \cap S_{c_j} = \emptyset$.

One of the most important concepts in coding theory - visualization on the next slide

Codeword spheres



d=2e+1=3; e=1

Spheres of radius 1

Proof of the error correcting theorem

Proof (cont.) Suppose $\mathbf{x} \in S_{c_i} \cap S_{c_j}$, then $d(\mathbf{x}, \mathbf{c_i}) \leq e$ and $d(\mathbf{x}, \mathbf{c_j}) \leq e$.
Using triangle inequality

$$d(\mathbf{x}, \mathbf{c_i}) + d(\mathbf{x}, \mathbf{c_j}) \geq d(\mathbf{c}_i, \mathbf{c_j}) \Rightarrow d(\mathbf{c}_i, \mathbf{c_j}) \leq 2e$$

Contradiction as $d(\mathbf{c}_i, \mathbf{c_j}) \geq 2e + 1$, so $S_{c_i} \cap S_{c_j} = \emptyset$.
If $t \leq e$ errors are introduced and $\mathbf{c_i}$ transmitted then $r \in S_{c_i}$.

• For error detection at least $2e + 1$ errors turns a codeword into another one. Therefore, up to $2e$ errors can always be detected.

– The case the min. distance is even $d = 2e$ is very similar (exercise 7)

Example

Example Assume $d = 4$ is even, and consider the codewords (of some code)

$$\mathbf{c}_1 = (110011) \quad \mathbf{c}_2 = (001111)$$

If the received word is (101000) then the decoder cannot decide whether $\mathbf{c}_1$ or $\mathbf{c}_2$ was sent.

The received word not in the spheres of radius 1 !

Detection is clearly possible - simply $\mathbf{r}$ is not a codeword.

Combining detection and correction

Theorem If C is an $[n, M]$ code with min. distance d. Then C can correct up to $\lfloor (d-1)/2 \rfloor$ errors. If used for error detection only, C can detect $d-1$ errors.

- Even case and odd case are different when both correction and detection are performed !

Theorem If C is an $[n, M]$ code with min. distance $d = 2e + 1$. Then C can correct up to e errors but cannot simultaneously detect additional errors.

Proof Decoder can correct up to e errors (and detect) but if $e+1$ errors occurs then $S_{c_i} \to S_{c_j}$ and no detection.

Decoding example

Consider the code C (example 6) with codewords:

$$\mathbf{c}_1 = (00000), \mathbf{c}_2 = (10110), \mathbf{c}_3 = (01011), \mathbf{c}_4 = (11101)\}$$

If we would construct the spheres of radius 1 (since $d = 3$)

$$\begin{aligned}
S_{c_1} &= \{(00000), (10000), (01000), (00100), (00010), (00001)\} \\
S_{c_2} &= \{(10110), (00110), (11110), (10010), (10100), (10111)\} \\
S_{c_3} &= \{(01011), (11011), (00011), (01111), (01001), (01010)\} \\
S_{c_4} &= \{(11101), (01101), (10101), (11001), (11111), (11100)\}
\end{aligned}$$

The set of vectors that are not in spheres is,

$$S^* = \{(11000), (01100), (10001), (00101), (01110), (00111), (10011), (11010)\}.$$

Decoding example II

$\mathbf{c}_1 = (00000), \mathbf{c}_2 = (10110), \mathbf{c}_3 = (01011), \mathbf{c}_4 = (11101)\}$

- Let $\mathbf{r}$=(00011). Then we compute,

  $$d(\mathbf{c}_1, \mathbf{r}) = 2, \quad d(\mathbf{c}_2, \mathbf{r}) = 3, \quad d(\mathbf{c}_3, \mathbf{r}) = 1, \quad d(\mathbf{c}_4, \mathbf{r}) = 4,$$

  Decode as $\mathbf{c}_3$.

- Let $\mathbf{r} = (11000) \in S^*$. Then we compute,

  $$d(\mathbf{c}_1, \mathbf{r}) = 2, \quad d(\mathbf{c}_2, \mathbf{r}) = 3, \quad d(\mathbf{c}_1, \mathbf{r}) = 3, \quad d(\mathbf{c}_1, \mathbf{r}) = 2.$$

  Cannot decode, the receiver knows there are at least 2 errors.

Decoding - combining correction and detection

$\mathbf{c}_1 = (00000), \mathbf{c}_2 = (10110), \mathbf{c}_3 = (01011), \mathbf{c}_4 = (11101)\}$

- Last case: Suppose $\mathbf{c}_1$ is sent and 2 errors are present so that

  $$\mathbf{r} = (10100).$$

  – Receiver decides in favour of $\mathbf{c}_2$ (closest) - makes error.

  – But **cannot detect 2 errors** if used at the same time for error correcting (only one error; distance to $\mathbf{c}_2$ is 1).

  – Without correcting can detect 2 errors.

Decoding complexity

- Important to design error correcting capability related to a given application.
- If M is large, say $M = 2^{50}$ it is infeasible to find the closest codeword ! 10^6 distance computations/sec gives 20 years for a single error correction.
- Also computing min. distance $\approx M^2/2$ is infeasible.
- Another issue is the efficiency (rate) of the code - e.g. given n and d (desired) how do we maximize k ?
- Also given n and k how do we maximize d ?

Shannon's theorem- Introduction

Assume we toss a coin and want to transmit the information by tel. wire. Further assumptions:

- Have two different symbols 0 and 1 as our alphabet symbols
- The coin is tossed t times per minute and the channel can handle $2t$ tosses per minute.
- Channel is noisy with probability of error $p = Pb(1 \rightarrow 0) = Pb(0 \rightarrow 1)$.

No restriction on the channel but **need arbitrary small probability of error probability after decoding**.

Idea: use repetition code of large length N.

Shannon's theorem- Preparation

- Then if $p = 0.001$ the decoder makes an error with:

$$P_e = \sum_{0 \le k < N/2} \binom{N}{k} (1-p)^k p^{N-k} < (0.07)^N,$$

thus $P_e \to 0$ for $N \to \infty$.

Problem - can only send 2 symbols for each tossing! SOLUTION?

- YES, one of the greatest result in coding/information theory due to C. Shannon, 1948.

Shannon's theorem- Notation

Suppose we use $C = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$, $|\mathbf{x}_i| = n$ and the maximum likelihood decoding.

Lep P_i - the probability of making incorrect decision given $\mathbf{x}_i$ is transmitted.

$$P_C := \frac{1}{M} \sum_{i=1}^{M} P_i \quad \text{prob. of incorrect decoding of word}$$

- Consider all possible codes with given parameters and define:

$$P^*(M, n, p) := \min_C P_C$$

Shannon's theorem

Theorem If the rate $R = \frac{\log_2 M}{n}$ is in the range $0 < R < 1 - H(p)$ and $M_n := 2^{\lfloor Rn \rfloor}$ then

$$P^*(M_n, n, p) \to 0 \text{ if } n \to \infty$$

Comments: Crucial dependence on p through the binary entropy function

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

– Properties of H:

$$H(0) = H(1) = 0 \text{ and } \max_p H(p) = 1 \text{ for } p = 1/2.$$

– Number of errors in received word is random var. with mean value np and variance $np(1-p)$.

Shannon's theorem - interpretation

– First note that the capacity of a BSC is,

$$C_{BSC} = 1 - H(p).$$

Two interesting cases (though rate is fixed):

- $p \to 0 \Rightarrow H(p) \to 0 \Rightarrow C_{BSC} \to 1$. To achieve $R \approx 1$ almost no redundancy (parity bits) as $M = 2^{\lfloor Rn \rfloor} \approx 2^n$
- $p \to 1/2 \Rightarrow H(p) \to 1 \Rightarrow C_{BSC} \to 0$. To achieve $R > 0$ redundancy (parity bits) as M is small (few information bits)

– Observe that proof is nonconstructive - no procedure how to design such a code.

Proof of Shannon's theorem

OPTIONAL FOR INTERESTED STUDENTS

Shannon's theorem - some estimates

w :=– the number of errors in the received word

$b := (np(1-p)/(\epsilon/2))^{1/2}$

Then,

$$P(w > np + b) \leq \frac{1}{2}\epsilon \quad \text{Chebyshev's inequality}$$

– Since $p < 1/2$ then $\rho := \lfloor np + b \rfloor < n/2$ for large n

– If $B_\rho(\mathbf{x}) = \{\mathbf{y} : d(\mathbf{x}, \mathbf{y}) \leq \rho\}$ is a sphere of radius ρ then,

$$|B_\rho(\mathbf{x})| = \sum_{i \leq \rho} \binom{n}{i} < \frac{1}{2} n \binom{n}{\rho} \leq \frac{1}{2} n \frac{n^n}{\rho^\rho (n-\rho)^{n-\rho}}$$

Need some more estimates:)

Shannon's theorem - some estimates II

$$\begin{aligned} \frac{\rho}{n}\log\frac{\rho}{n} &= p\log p + O(n^{-1/2}) \\ (1-\frac{\rho}{n})\log(1-\frac{\rho}{n}) &= q\log q + O(n^{-1/2})(n\to\infty) \end{aligned}$$

- Finally need two functions. If $\mathbf{u},\mathbf{v},\mathbf{y}\in\{0,1\}^n$, $\mathbf{x}\in C$ then

$$f(\mathbf{u},\mathbf{v}) = \begin{cases} 0, & \text{if } d(\mathbf{u},\mathbf{v}) > \rho \\ 1, & \text{if } d(\mathbf{u},\mathbf{v}) \le \rho \end{cases}$$

$$g_i(\mathbf{y}) = 1 - f(\mathbf{y},\mathbf{x}_i) + \sum_{j\neq i} f(\mathbf{y},\mathbf{x}_j).$$

FACT: If $\mathbf{x}_i$ is unique codeword s.t. $d(\mathbf{x}_i,\mathbf{y})\le\rho$ then $g_i(\mathbf{y})=0$, and $g_i(\mathbf{y})\ge 1$ otherwise.

Shannon's theorem - proof

Proof: We pick the codewords $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_M$ at random.

Decoding: If only $\mathbf{x}_i$ is s.t. $d(\mathbf{x}_i,\mathbf{y})\le\rho$ then decode $\mathbf{y}$ as $\mathbf{x}_i$, otherwise decode as say $\mathbf{x}_1$ (max. likelihood decoding) .

- Express P_i using g_i,

$$\begin{aligned} P_i &= \sum_{\mathbf{y}\in\{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i)g_i(\mathbf{y}) \quad (\mathbf{x}_i \text{ is fixed }) \\ &= \underbrace{\sum_{\mathbf{y}\in\{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i)\{1-f(\mathbf{y},\mathbf{x}_i)\}}_{Pb(\mathbf{y}\notin B_\rho(\mathbf{x}_i))} + \sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y}|\mathbf{x}_i)f(\mathbf{y},\mathbf{x}_j). \end{aligned}$$

Using $P(w>np+b)=P(w>\rho)\le\frac{1}{2}\epsilon$ we get (next page)

Shannon's theorem - proof II

$$P_c \leq \frac{1}{2}\epsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i} P(\mathbf{y},\mathbf{x}_i)f(\mathbf{y},\mathbf{x}_i)$$

- Now we use the fact that $P^*(M,n,p) < \mathcal{E}(P_C)$, where $\mathcal{E}(P_C)$ is expected value over all possible codes C. Hence,

$$\begin{aligned} P^*(M,n,p) &\leq \frac{1}{2}\epsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i}\mathcal{E}(P(\mathbf{y},\mathbf{x}_i))\mathcal{E}(f(\mathbf{y},\mathbf{x}_i)) \\ &= \frac{1}{2}\epsilon + M^{-1}\sum_{i=1}^{M}\sum_{\mathbf{y}}\sum_{j\neq i}\mathcal{E}(P(\mathbf{y},\mathbf{x}_i))\cdot\frac{|B_\rho|}{2^n} \\ &= \frac{1}{2}\epsilon + (M-1)2^{-n}|B_\rho|. \end{aligned}$$

Shannon's theorem - proof III

Finally, we take logs, apply our estimates and divide by n to get,

$$\begin{aligned} &n^{-1}\log(P^*(M,n,p) - \frac{1}{2}\epsilon) \\ &\leq \underbrace{n^{-1}\log M - (1 + p\log p + q\log q)}_{R-(1-H(p))<0} + O(n^{-1/2}). \end{aligned}$$

This leads to,

$$n^{-1}\log(P^*(M_n,n,p) - \frac{1}{2}\epsilon) < -\beta < 0,$$

for $n \geq n_0$, i.e. $P^*(M_n,n,p) < \frac{1}{2}\epsilon + 2^{-\beta n}$.

Chapter 3

Coding theory

Contents of the chapter:

- Decoding
- Shannon
- Vector spaces
- Linear codes
- Generator matrix
- Parity check

Example

Example Assume $d = 4$ is even, and consider the codewords

$$\mathbf{c}_1 = (110000) \quad \mathbf{c}_2 = (001100)$$

If the received word is (101000) then the decoder cannot decide whether $\mathbf{c}_1$ or $\mathbf{c}_2$ was sent.

The received word not in the spheres of radius 1 !

Detection is clearly possible - simply $\mathbf{r}$ is not a codeword.

Combining detection and correction

Theorem If C is an $[n, M]$ code with min. distance d. Then C can correct up to $\lfloor (d-1)/2 \rfloor$ errors. If used for error detection only, C can detect $d - 1$ errors.

- Even case and odd case are different when both correction and detection are performed !

Theorem If C is an $[n, M]$ code with min. distance $d = 2e + 1$. Then C can correct up to e errors but cannot simultaneously detect additional errors.

Proof Decoder can correct up to e errors (and detect) but if $e + 1$ errors occurs then $S_{c_i} \rightarrow S_{c_j}$ and no detection.

Decoding example

Consider the code C (example 6) with codewords:

$$\mathbf{c}_1 = (00000), \mathbf{c}_2 = (10110), \mathbf{c}_3 = (01011), \mathbf{c}_4 = (11101)\}$$

If we would construct the spheres of radius 1 (since $d = 3$)

$$\begin{aligned} S_{c_1} &= \{(00000), (10000), (01000), (00100), (00010), (00001)\} \\ S_{c_2} &= \{(10110), (00110), (11110), (10010), (10100), (10111)\} \\ S_{c_3} &= \{(01011), (11011), (00011), (01111), (01001), (01010)\} \\ S_{c_4} &= \{(11101), (01101), (10101), (11001), (11111), (11100)\} \end{aligned}$$

The set of vectors that are not in spheres is,

$$S^* = \{(11000), (01100), (10001), (00101), (01110), (00111), (10011), (11010)\}.$$

Decoding example II

- Let $\mathbf{r}$=(00011). Then we compute,

  $$d(\mathbf{c}_1, \mathbf{r}) = 2, \quad d(\mathbf{c}_2, \mathbf{r}) = 3, \quad d(\mathbf{c}_1, \mathbf{r}) = 4, \quad d(\mathbf{c}_1, \mathbf{r}) = 2,$$

  Decode as $\mathbf{c}_3$.

- Let $\mathbf{r} = (11000) \in S*$. Then we compute,

  $$d(\mathbf{c}_1, \mathbf{r}) = 2, \quad d(\mathbf{c}_2, \mathbf{r}) = 3, \quad d(\mathbf{c}_1, \mathbf{r}) = 3, \quad d(\mathbf{c}_1, \mathbf{r}) = 2.$$

  Cannot decode, the receiver knows there are at least 2 errors.

- Suppose $\mathbf{c}_1$ is sent and 2 errors are present so that $\mathbf{r} = (10100)$. Receiver decides in favour of $\mathbf{c}_2$ (closest).

  – But cannot detect 2 errors if used for error correcting. Without correcting can detect 2 errors.

Decoding complexity

- Important to design error correcting capability related to a given application.
- If M is large, say $M = 2^{50}$ it is infeasible to find the closest codeword !
- Also computing min. distance $\approx M^2/2$ is infeasible.
- Another issue is the efficiency (rate) of the code - e.g. given n and d (desired) how do we maximize k ?
- Also given n and k how do we maximize d ?

Shannon's theorem- Introduction

Assume we toss a coin and want to transmit the information by tel. wire. Further assumptions:

- Have two different symbols 0 and 1 as our alphabet symbols
- The coin is tossed t times per minute and the channel can handle $2t$ tosses per minute.
- Channel is noisy with probability of error $p = Pb(1 \to 0) = Pb(0 \to 1)$.

No restriction on the channel $\Rightarrow$ arbitrary small probability of error decoding.

Idea: use repetition code of large length N.

Shannon's theorem- Preparation

- Then if $p = 0.001$ the decoder makes an error with:

$$P_e = \sum_{0 \le k < N/2} \binom{N}{k} (1-p)^k p^{N-k} < (0.07)^N,$$

thus $P_e \to 0$ for $N \to \infty$.

Problem - can only send 2 symbols for each tossing! SOLUTION?

- YES, one of the greatest result in coding/information theory due to C. Shannon, 1948.

Decoding Shannon Vector spaces Linear codes Generator matrix Parity check

Shannon's theorem- Notation

Suppose we use $C = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$, $|\mathbf{x}_i| = n$ and the maximum likelihood decoding.

Lep P_i - the probability of making incorrect decision given $\mathbf{x}_i$ is transmitted.

$$P_C := \frac{1}{M} \sum_{i=1}^{M} P_i \quad \text{prob. of incorrect decoding of word}$$

- Consider all possible codes with given parameters and define:

$$P^*(M, n, p) := \min_C P_C$$

Shannon's theorem

Theorem If the rate $R = \frac{\log_2 M}{n}$ is in the range $0 < R < 1 - H(p)$ and $M_n := 2^{\lfloor Rn \rfloor}$ then

$$P^*(M_n, n, p) \to 0 \text{ if } n \to \infty$$

Comments: Crucial dependence on p through the binary entropy function

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

– Properties of H:

$$H(0) = H(1) = 0 \text{ and } \max_p H(p) = 1 \text{ for } p = 1/2.$$

– Number of errors in received word is random var. with mean value np and variance $np(1-p)$.

Shannon's theorem - interpretation

– First note that the capacity of a BSC is,

$$C_{BSC} = 1 - H(p).$$

Two interesting cases (though rate is fixed):

- $p \to 0 \Rightarrow H(p) \to 0 \Rightarrow C_{BSC} \to 1$. To achieve $R \approx 1$ almost no redundancy (parity bits) as $M = 2^{\lfloor Rn \rfloor} \approx 2^n$
- $p \to 1/2 \Rightarrow H(p) \to 1 \Rightarrow C_{BSC} \to 0$. To achieve $R > 0$ redundancy (parity bits) as M is small (few information bits)

– Observe that proof is nonconstructive - no procedure how to design such a code.

Motivation for linear codes

- A class of codes with nice algebraic structure.
- Not always the best ones but allows for efficient coding and decoding.
- Additional structural constraints gives families of cyclic and BCH codes
- Hamming codes are typical representative, but many other good codes Reed-Muller, Hadamard codes etc.

Code as a vector space

Need to formally define the main parameters

- Alphabet A - finite field with q elements, e.g. $A = GF(2)$ then $|A| = 2$ or $A = GF(p^r)$ so $|A| = p^r$.
- Message space - the set of all k-tuples over F, denoted $V_k(F)$. In total q^k messages.
- The message k-tuples embedded into n-tuples, $n \geq k$. Redundancy used in error correction/detection.
- One-to-one correspondence

$$q^k \text{ messages } \leftrightarrow q^k \;\; n- \text{ tuples in } V_k(F)$$

Question: Can we choose q^k n-tuples so that they form a k dim. *subspace* in $V_n(F)$?

Vector spaces-basics

- What is a k-dim. vector subspace $S \subset V_n(F)$?
- Simply, subspace is determined by k linearly independent vectors in $V_n(F)$

  Example Recall our code $C = \{00000, 10110, 01011, 11101\}$. Then any two vectors in $C \setminus \{\mathbf{0}\}$ are linearly independent. E.g. taking as basis $\mathbf{c_1} = 10110, \mathbf{c_2} = 01011$ we get C as,

  $$C = a_1 c_1 + a_2 c_2, (a_1, a_2) \in F;\ F = GF(2^2)$$

  Three different basis (six up to permutation), same code !
- In general, the number of selecting k lin. ind. vectors is

$$(q^n - 1)(q^n - q)(q^n - q^2) \cdots (q^n - q^{k-1}) = \prod_{i=0}^{k-1} (q^n - q^i).$$

Counting subspaces

- Each k-dimensional subspace contains

$$(q^k - 1)(q^k - q)(q^k - q^2) \cdots (q^k - q^{k-1}) = \prod_{i=0}^{k-1} (q^k - q^i)$$

  ordered sets of k linearly independent vectors.
- The total number of k-dimensional subspaces in $V_n(F)$ is,

$$\frac{\prod_{i=0}^{k-1} (q^n - q^i)}{\prod_{i=0}^{k-1} (q^k - q^i)}$$

  Example In our case $q = 2$, $n = 5$, $k = 2$

$$\prod_{i=0}^{k-1} (q^n - q^i) = \prod_{i=0}^{1} (2^5 - 2^i) = 31 \cdot 30 = 960.$$

Counting subspaces II

$$\prod_{i=0}^{k-1}(q^k - q^i) = \prod_{i=0}^{1}(2^2 - 2^i) = 3 \cdot 2 = 6; \quad \frac{\prod_{i=0}^{k-1}(q^n - q^i)}{\prod_{i=0}^{k-1}(q^k - q^i)} = \frac{960}{6} = 160.$$

Where does this 6 comes from ?

$$\begin{array}{ll} (10000), (01000) & (01000), (10000) \\ (11000), (01000) & (01000), (11000) \\ (11000), (10000) & (10000), (11000) \end{array}$$

All gives the same subspace !

Basis of a code

- We can select any of the 160 subspaces to construct linear $[5, 2]$ code C.
- But need a correspondence between subspace and the message space ?
- Let us select a basis $B = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ of S (k-dim. subspace of $V_n(F)$) and define,

$$f : M \to S; \quad f(\mathbf{m}) = \sum_{i=1}^{k} m_i \mathbf{v}_i,$$

where $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ is a message k-tuple, $\mathbf{m} \in M$.

Constructing linear code - example

Example Let $M = \{(00), (10), (01), (11)\}$

- Define subspace S of $V_4(Z_2)$ through the basis $B = \{\mathbf{v}_1, \mathbf{v}_2\}$,

$$\mathbf{v}_1 = (1100) \quad \mathbf{v}_1 = (0110).$$

- Then f maps M to S as follows,

$$\begin{aligned}(00) &\rightarrow (0000)\\(10) &\rightarrow (1100)\\(01) &\rightarrow (0110)\\(11) &\rightarrow (1010)\end{aligned}$$

Thus $S = C = \{(0000), (1100), (0110), (1010)\}$.

Selecting a "good" subspace

- Many choices for the subspace (linear code) for fixed n, k. E.g.

$$\begin{aligned}B &= \{(10000), (01000)\} \Rightarrow d_C = 1,\\B &= \{(10110), (01011)\} \Rightarrow d_C = 3,\\B &= \{(10111), (11110)\} \Rightarrow d_C = 2,\end{aligned}$$

- Choose the subspace with largest Hamming distance.
- For fixed k can increase n - more check digits (greater potential for error correcting). But smaller rate typical trade-off.

Definition A linear (n, k)-code is a k-dimensional subspace of $V_n(F)$.

Minimum distance of linear codes

- For a nonlinear $[n, M]$ code computing d requires computing $\binom{M}{2}$ Hamming distances. Linear code is easier to handle !

  Definition The Hamming weight of $\mathbf{v} \in V_n(F)$ is the number of nonzero coordinates in $\mathbf{v}$, i.e.

  $$w(v) = \#\{v_i \neq 0, 1 \leq i \leq n\}$$

  Definition The Hamming weight of an (n, k) code C is,

  $$w(C) = \min\{w(\mathbf{x}) : \mathbf{x} \in C, \mathbf{x} \neq 0\}.$$

- If $C = \{(0000), (1100), (0011), (1111)\}$ then the Hamming distance of the code equals to the Hamming weight of the code !

Hamming weight of a linear code

Theorem: Let d be the distance of an (n, k) code C. Then,

$$d = w(C)$$

Proof: By definition, $d = \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}$. Also

$$d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$$

But $\mathbf{x} - \mathbf{y} \in C$ (C is subspace) so,

$$d = \min\{w(\mathbf{z}) : \mathbf{z} \in C, \mathbf{z} \neq 0\}$$

- Computing the distance equivalent to finding codeword with max number of zeroes !

Representing linear codes

- It is of interest (for decoding) to select particular basis.

Example: Let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ be a basis of a $(5,3)$ code. Define,

$$G = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- If $\mathbf{m} = (m_1 m_2 m_3) \in M$ then

$$c = \mathbf{m}G = m_1\mathbf{v}_1 + m_2\mathbf{v}_2 + m_3\mathbf{v}_3$$

  – E.g. $\mathbf{m} = (101)$ then $\mathbf{m}G = (01101)$.
- Selecting the basis $\mathbf{u}_1 = (10000), \mathbf{u}_2 = (01010), \mathbf{u}_3 = (00111)$ (same code) $\mathbf{m}G' = (10111)$.

Generator matrix of code

Definition: A generator matrix G of an (n,k)-code C is a $k \times n$ matrix whose rows are a vector space basis for C.

- Codewords of C = linear combinations of rows of G.
- Generator matrix G not unique - elementary row operations gives the same code
- Would like to find a generator matrix in standard form,

$$G = [I_k \;\; A]$$

$$I_k \text{ identity } k \times k; A \; - \; k \times (n-k) \text{ matrix}$$

– Can we for a given C always find G in a standard form ? NO, but we can find equivalent code !

Equivalent codes

- Main idea: permuting coordinates of the codewords does not affect Hamming weight !

$$
\begin{aligned}
C &= \{(0000),(1100),(0011),(1111)\} \\
C' &= \{(0000),(0110),(1001),(1111)\}
\end{aligned}
$$

- We can get *equivalent* code (not necessarily identical) !

Definition Two (n,k)-codes C and C' are said to be equivalent if there exists a *permutation matrix* P such that $G' = GP$.

- P permutes the columns of G (coordinates of the codewords)

Theorem If C is an (n,k)-code over F then there exists G for C or for an equivalent code C' such that $G = [I_k A]$.

Transforming the code - example

- Want to transform C into C' (equivalent not identical codes)

$$
\tilde{G} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} ; \; G' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}
$$

- Step 1: $\tilde{G} \to G$ (add row 1 to rows 2 and 3)

$$
\tilde{G} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}
$$

- Step 2: $G' = GP$ (interchange columns 1 and 3)

$$
\tilde{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Orthogonal spaces

– Define inner product of $\mathbf{x}, \mathbf{y} \in V_n(F)$,

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

- Remark that $\mathbf{x} \cdot \mathbf{x} = 0 \Rightarrow \mathbf{x} = 0$ if $\mathbf{x} \in \mathbb{R}$. But not the case when F is a finite field. E.g.

$$\mathbf{x} = (101) \Rightarrow \mathbf{x} \cdot \mathbf{x} = 1 + 0 + 1 = 0$$

Orthogonal vectors if $\mathbf{x} \cdot \mathbf{y} = 0$.

Definition Let C be an (n, k) code over F. The orthogonal complement of C (dual code of C) is

$$C^{\perp} = \{\mathbf{x} \in V_n(F) : \mathbf{x} \cdot \mathbf{y} = 0 \text{ for all } \mathbf{y} \in C\}$$

Dual code

Theorem 3.3. If C is an (n, k) code over F then $C^{\perp}$ is an $(n, n-k)$ code over F.

Proof (see the textbook). First show that $C^{\perp}$ is a subspace of $V_n(F)$, then show that $dim(C^{\perp}) = n - k$.

- What is a generator matrix of $C^{\perp}$?

Corollary 3.4 If $G = [I_k A]$ is a generator matrix of C then $H = [-A^T I_{n-k}]$ is a generator matrix of $C^{\perp}$!

Proof We have, $GH^T = I_k(-A) + AI_{n-k} = 0$, i.e. rows of H orthogonal to rows of G. By definition $span(H) = C^{\perp}$

Dual code - example

Example Let C be an $(6,3)$ code defined by,

$$\tilde{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} = [I_3\ A],$$

Then,

$$H = [-A^T I_2] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Check that $GH^T = 0$, and linear independency of rows of H !

Parity check matrix

- The condition $GH^T = 0$ essentially means,

  $$\mathbf{c} \in C \Leftrightarrow H\mathbf{c}^T = 0.$$

  Comes from $mG = \mathbf{c}$ after multiplying with H^T

  Definition If H is a gen. matrix of $C^\perp$ then H is called a parity check matrix .

- But also if G is the generator matrix of C then it is parity check matrix for $C^\perp$.
- Easy transformation if standard form,

  $$G = [I_k A] \Leftrightarrow H = [-A^T I_{n-k}]$$

Parity check matrix II

- Can specify C given H (standard form), but no need to perform $H \to G \to c = mG$!
- Encoding of $\mathbf{m} = (m_1 m_2 \dots m_k)$ (in standard form is mapping of m to

$$\mathbf{c} = (m_1 m_2 \dots m_k x_1 x_2 \dots x_{n-k})$$

- The x_i are called check symbols - they provide redundancy to detect and correct errors !
- Given $\mathbf{m}$ and H the check symbols are determined through,

$$H\mathbf{c}^T = 0$$

Parity checks - example

Let C be a $(6,3)$ code and with the parity check matrix,

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

- Which codeword encodes the message $\mathbf{m} = (101)$?
  – Depend on the basis of C ! If we prefer standard form $(G = I_K A])$ then, $\mathbf{c} = (101x_1x_2x_3)$.

- Using $H\mathbf{c}^T = 0$ gives,

$$\begin{aligned} 1 + 1 + x_1 = 0 &\quad\to\quad x_1 = 0 \\ 1 + x_2 = 0 &\quad\to\quad x_2 = 1 \Rightarrow \mathbf{c} = (101011) \\ 1 + x_3 = 0 &\quad\to\quad x_3 = 1 \end{aligned}$$

Parity checks - example

- Easy to determine general equations for x_i,

$$\begin{aligned} m_1 + m_3 &= x_1 \\ m_1 + m_2 &= x_2 \\ m_2 + m_3 &= x_3 \end{aligned}$$

– Another way of computing the codewords is to use $H = [-A^T I_3]$,

$$G = [I_3 A] = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

and $\mathbf{c} = (101011)$

Properties of parity check matrix

Theorem Let C be an (n, k) code over F. Every set of $s - 1$ columns of H are linearly independent iff $w(C) \geq s$.

Proof $\Rightarrow$ Denote $H = [\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_n}]$- $\mathbf{h_i}$ columns of H.

– Assumption any $s - 1$ columns of H lin. independent. Then,

$$H\mathbf{c}^T = [\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_n}]\mathbf{c}^T = \sum_{i=1}^{n} c_i \mathbf{h_i} = 0$$

– If $wt(\mathbf{c}) \leq s - 1$, contradiction. Thus, $wt(\mathbf{c}) \geq s$. Since c is arbitrary we have $w(C) \geq s$.

Properties of parity check matrix II

Proof.
(cont.) $\Leftarrow$ Assume $w(C) \geq s$ and some set of $t \leq s-1$ columns of H are lin. dependent.

$$\exists \lambda_{i_j} \in F : \sum_{j=1}^{t} \lambda_{i_j} \mathbf{h}_{i_j} = 0$$

– Construct $\mathbf{c}$ s.t.

$$c_{i_j} = \begin{cases} \lambda_{i_j} & 1, \leq j \leq t \\ 0, & \text{otherwise} \end{cases}$$

- Legal codeword as $H\mathbf{c}^T = 0$, but $w(c) = t \leq s-1$, contradiction. REMARK: We can compute the distance of the code in this way !

$\square$

Shannon - proof

OPTIONAL - FOR INTERESTED STUDENTS

Shannon's theorem - some estimates

w :=– the number of errors in the received word

$b := (np(1-p)/(\epsilon/2))^{1/2}$

Then,

$$P(w > np + b) \leq \frac{1}{2}\epsilon \text{ Chebyshev's inequality}$$

– Since $p < 1/2$ then $\rho := \lfloor np + b \rfloor < n/2$ for large n

– If $B_\rho(\mathbf{x}) = \{\mathbf{y} : d(\mathbf{x},\mathbf{y}) \leq \rho\}$ is a sphere of radius ρ then,

$$|B_\rho(\mathbf{x})| = \sum_{i \leq \rho} \binom{n}{i} < \frac{1}{2} n \binom{n}{\rho} \leq \frac{1}{2} n \frac{n^n}{\rho^\rho (n-\rho)^{n-\rho}}$$

Need some more estimates:)

Shannon's theorem - some estimates II

$$\begin{aligned} \frac{\rho}{n} \log \frac{\rho}{n} &= p \log p + O(n^{-1/2}) \\ (1 - \frac{\rho}{n}) \log(1 - \frac{\rho}{n}) &= q \log q + O(n^{-1/2})(n \to \infty) \end{aligned}$$

- Finally need two functions. If $\mathbf{u}, \mathbf{v}, \mathbf{y} \in \{0,1\}^n$, $\mathbf{x} \in C$ then

$$f(\mathbf{u},\mathbf{v}) = \begin{cases} 0, & \text{if } d(\mathbf{u},\mathbf{v}) > \rho \\ 1, & \text{if } d(\mathbf{u},\mathbf{v}) \leq \rho \end{cases}$$

$$g_i(\mathbf{y}) = 1 - f(\mathbf{y},\mathbf{x}_i) + \sum_{j \neq i} f(\mathbf{y},\mathbf{x}_j).$$

FACT: If $\mathbf{x}_i$ is unique codeword s.t. $d(\mathbf{x}_i,\mathbf{y}) \leq \rho$ then $g_i(\mathbf{y}) = 0$, and $g_i(\mathbf{y}) \geq 1$ otherwise.

Shannon's theorem - proof

Proof: We pick the codewords $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ at random.

Decoding: If only $\mathbf{x}_i$ is s.t. $d(\mathbf{x}_i, \mathbf{y}) \leq \rho$ then decode $\mathbf{y}$ as $\mathbf{x}_i$, otherwise decode as say $\mathbf{x}_1$.

- Express P_i using g_i,

$$\begin{aligned} P_i &= \sum_{\mathbf{y}\in\{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i) g_i(\mathbf{y}) \;\; (\mathbf{x}_i \text{ is fixed }) \\ &= \underbrace{\sum_{\mathbf{y}\in\{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i)\{1 - f(\mathbf{y}, \mathbf{x}_i)\}}_{Pb(\mathbf{y}\notin B_\rho(\mathbf{x}_i))} + \sum_{\mathbf{y}} \sum_{j\neq i} P(\mathbf{y}|\mathbf{x}_i) f(\mathbf{y}, \mathbf{x}_i). \end{aligned}$$

Using $P(w > np + b) = P(w > \rho) \leq \frac{1}{2}\epsilon$ we get (next page)

Shannon's theorem - proof II

$$P_c \leq \frac{1}{2}\epsilon + M^{-1} \sum_{i=1}^{M} \sum_{\mathbf{y}} \sum_{j\neq i} P(\mathbf{y}, \mathbf{x}_i) f(\mathbf{y}, \mathbf{x}_i)$$

- Now we use the fact that $P^*(M, n, p) < \mathcal{E}(P_C)$, where $\mathcal{E}(P_C)$ is expected value over all possible codes C. Hence,

$$\begin{aligned} P^*(M, n, p) &\leq \frac{1}{2}\epsilon + M^{-1} \sum_{i=1}^{M} \sum_{\mathbf{y}} \sum_{j\neq i} \mathcal{E}(P(\mathbf{y}, \mathbf{x}_i)) \mathcal{E}(f(\mathbf{y}, \mathbf{x}_i)) \\ &= \frac{1}{2}\epsilon + M^{-1} \sum_{i=1}^{M} \sum_{\mathbf{y}} \sum_{j\neq i} \mathcal{E}(P(\mathbf{y}, \mathbf{x}_i)) \cdot \frac{|B_\rho|}{2^n} \\ &= \frac{1}{2}\epsilon + (M-1)2^{-n}|B_\rho|. \end{aligned}$$

Shannon's theorem - proof III

Finally, we take logs, apply our estimates and divide by n to get,

$$n^{-1}\log(P^*(M,n,p) - \frac{1}{2}\epsilon)$$
$$\leq \underbrace{n^{-1}\log M - (1 + p\log p + q\log q)}_{R-(1-H(p))<0} + O(n^{-1/2}).$$

This leads to,

$$n^{-1}\log(P^*(M_n,n,p) - \frac{1}{2}\epsilon) < -\beta < 0,$$

for $n \geq n_0$, i.e. $P^*(M_n,n,p) < \frac{1}{2}\epsilon + 2^{-\beta n}$.

Chapter 4

Decoding of linear codes and MacWilliams identity

Contents of the chapter:

- Reminder
- Hamming
- Group theory
- Standard array
- Weight distribution
- MacWilliams identity

Linear codes - repetition

- Linear code (n, k) is a linear subspace of $V_n(A)$ of dimension k.
- Specified by the generator matrix G (alternatively parity check matrix H)
$$GH^T = 0.$$
- Comes easily from $H\mathbf{c}^T = 0$ for any codeword $\mathbf{c} \in C$.
- $G = [I_k \ A]$ in standard form was particularly suitable.

Standard form - repetition

- Could not always find G in standard form by elementary row operations !
- Examples (better)

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} G' = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Solution: Find equivalent code - permutation of columns allowed.

Main result - reminder

Theorem Let C be an (n, k) code over F. Every set of $s-1$ columns of H are linearly independent iff $w(C) \geqslant s$.

- Special case is $s = 3$ - no 2 columns of H are linearly dependent

Constructing single-error correcting code

Need a code with $w(C) = 3$

- Previous result states that we need H s.t. no 2 or fewer columns of H are lin. dependent !
- SOLUTION: Do not use all-zero vector and no column is a scalar multiple of other column.

– The construction procedure is:

- Find H with no lin. dependency of any two columns (easy)
- For explicit definition of C we need a generator matrix G, i.e. $H \to G$.

–Special case when code alphabet is binary !

Example of single-error correcting code

Example Want to construct a single-error correcting $(7, 4)$ code ?

- Just ensure there is no repeated columns in H.
- Since G is a 4×7 matrix H is a 3×7 binary matrix
- Only one option (up to permutation of columns),

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

- Any other ordering of columns gives equivalent code.

– Can we construct 2-error correcting code in this way ?
YES, but a more complicated procedure (see the textbook).

What about a $(7, 5, 3)$ code ?

Hamming codes

- Single-error correcting codes; easy coding and decoding.

Definition A Hamming code of order r over $GF(q)$ is,
– an (n, k) code
– length of the code is $n = (q^r - 1)/(q - 1)$
– dimension $k = n - r$
– parity check matrix H_r of size $r \times n$ s.t. no 2 columns are lin. dependent.

- All the codes of min. distance 3; codewords have a maximum length, i.e. cannot increase the number of columns of H !

Due to $n = (q^r - 1)/(q - 1)$ cannot add any more columns to H

Binary Hamming codes

- Specified by a single parameter r.

Definition A binary Hamming code of order r is,

- an $(n, n-r)$ code
- length of the code is $n = 2^r - 1$
- dimension $k = n - r$
- parity check matrix H_r of size $r \times n$ s.t. all columns are distinct and nonzero.
- $d = 3$

Setting $r = 3$ we get a (7,4,3) Hamming code.

Perfect codes

Hamming codes (binary) example of perfect codes $(7, 4, 3)$, $(15, 11, 3)$ ...

Definition A perfect code is an e error-correcting $[n, M]$ code over A such that **every** n-tuple is in the sphere of radius e of some codeword.

Example Consider the vector space $V_7(2)$ - a set of binary vectors of length 7.

– There are $2^7 = 128$ vectors

– Each sphere of radius 1 contains 7+1=8 vectors

– 16 spheres cover the whole space $16 \times 8 = 128$

– Dimension of the code is $k = 4$, i.e. Hamming (7,4,3) code

Perfect codes II

- Spheres not only disjoint but exhaust the whole space A^n !

To see that Hamming codes are perfect observe,
– d(C)=3 thus $e = 1$; each sphere contains $1 + n(q-1)$ vectors

– the number of spheres is

$$q^k = q^{n-r}$$

(nmb. of codewords)

– so the spheres contain

$$[1 + n(q-1)]q^{n-r} = [1 + (q^r - 1)]q^{n-r} = q^n.$$

Decoding single-error correcting codes

- Need the concept of an error vector **e**,

$$\mathbf{r} = \mathbf{c} + \mathbf{e},$$

  where **r** is a received word.
- If H is a parity check of C and r is received then,

$$H\mathbf{r}^T = H(\mathbf{c}+\mathbf{e})^T = \underbrace{H\mathbf{c}^T}_{=0} + H\mathbf{e}^T = H\mathbf{e}^T$$

- We can easily deal with the cases $wt(\mathbf{e}) \leqslant 1$:

– If $\mathbf{e} = 0$ then $H\mathbf{e}^T = 0$ and accept **r** as transmitted codeword. – If $wt(\mathbf{e}) = 1$, say $e_i = \alpha \neq 0$ then $H\mathbf{e}^T = \alpha\mathbf{h}_i$

Decoding procedure (single-error)

H parity check matrix and $\mathbf{r}$ the received vector

1. Compute $H\mathbf{r}^T$
2. If $H\mathbf{r}^T = 0$ accept $\mathbf{r}$ as transmitted codeword
3. $H\mathbf{r}^T = \mathbf{s}^T \neq 0$ compare $\mathbf{s}^T$ with columns of H
4. If $\mathbf{s}^T = \alpha\mathbf{h}_i$ for some $1 \leqslant i \leqslant n$ then $\mathbf{e} = (0, 0, \ldots, \underbrace{\alpha}_{i}, 0, \ldots, 0)$; correct $\mathbf{r}$ to $\mathbf{c} = \mathbf{r} - \mathbf{e}$

Decoding - example

Let again

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Is $\mathbf{c}$=(1111111) a valid codeword ?

Assume $\mathbf{c}$=(1111111) is sent and $\mathbf{r}$=(0111111) received

Decode by computing $H\mathbf{r}^T = \mathbf{s}^T = 100^T$ - sum of last 6 columns of H.

Correct $\mathbf{r} \leftarrow \mathbf{r} + (1000000)$.

Reminder on group theory

- *Group* is a set G together with an operation "$\circ$" satisfying:
  1. $\forall a, b \in G : a \circ b \in G$ Algebraic closure
  2. $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$ Associativity
  3. $\exists! e \in G : \forall a \in G : a \circ e = e \circ a = a$ e is identity element
  4. $\forall a \in G, \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$ Inverse element
- $(G, \circ)$ is called Abelian if for all $a, b \in G$, $a \circ b = b \circ a$

Example of Groups

- $(\mathbb{Z}, +)$ is a group under usual integer addition. We check,

$$\forall a \in \mathbb{Z}, a + 0 = a; \quad a + (-a) = 0$$

- $(\mathbb{Z}, \cdot)$ is not a group as,

$$3^{-1} =? \ \text{i.e.} \quad 3 \cdot x = 1 \quad \text{has no solution in } \mathbb{Z}$$

- $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus 0 = \{1, 2, \dots, p-1\}$ is a group under multiplication $\pmod{p}$ iff p is prime.
- For example, $(\mathbb{Z}_5^*, \cdot \pmod{5})$ is a group since,

$$1^{-1} = 1; \ 2^{-1} = 3; \ 3^{-1} = 2; \ 4^{-1} = 4;$$

Structure of Groups

- A group G is *cyclic* if there exists a *generator* a of the group s.t.
$$\forall g \in G, \exists i \geqslant 0 : g = a^i = \overbrace{a \circ a \cdots \circ a}^{i \text{ times}} .$$
- 2 is a generator of $(\mathbb{Z}_5^*, \cdot \pmod 5)$ since,
$$2^0 = 1; \;\; 2^1 = 2; \;\; 2^2 = 4; \;\; 2^3 = 3 \pmod 5$$
- On the other hand 4 is not a generator as,
$$4^0 = 1; \;\; 4^1 = 4; \;\; 4^2 = 1 \pmod 5$$

Reminder on group theory II

We need the concepts of a subgroup, cosets and Lagrange theorem

- Let G be a group and $H \subset G$. H is called a *subgroup* of G if H is itself a group.

Definition Let H be a subgroup of G. The subset,
$$a \circ H = \{a \circ h \mid h \in H\}$$
is called the *left coset* of H containing a.

Theorem [Lagrange] For a subgroup H of G we have $\#H | \#G$.

Proof Show that $a \neq a'$ s.t. $a \notin a' \circ H$ then $(a \circ H) \cap (a' \circ H) = \emptyset$ and $\#(a \circ H) = \#H$.

Splitting the group into cosets

Group can be viewed as a union of cosets

Example Let $G = [(00), (10), (01), (11)]$ be a group with the group operation vector addition mod2.

Let $H = [(00), (10)] \subset G$. The cosets of H are,

$$H + (00) = H \quad H + (01) = [(01), (11)] = H + (11).$$

Thus $G = H \cup H + (01)$.

- The idea of standard array decoding is to think of C as a subgroup of order q^k in the group $V_n(F)$.
- Splitting $V_n(F)$ into cosets gives a convenient way of decoding any linear code.

Standard array decoding

Notation (vector addition): 0 is neutral element, inverse of a is $-a$

- A code C of size q^k and length n has $t = q^n/q^k = q^{n-k}$ cosets.
- These cosets are denoted $C_0, C_1, \ldots, C_{t-1}$, where $C_0 = C$.
- For each C_i let $\mathbf{l}_i$ (a coset leader), $0 \leqslant i \leqslant t-1$, be a vector of minimum weight in C_i

IDEA: Construct a $q^{n-k} \times q^k$ array S where $s_{i+1,j+1} = \mathbf{l}_i + \mathbf{c}_j$.

Entries in row $i + 1$ are elements of C_i and the first column contains coset leaders.

Standard array - example

For the binary $(5, 2)$ code with generator matrix,

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

the standard array is given by,

coset leaders				
00000	10101	01110	11011	codewords
00001	10100	01111	11010	
00010	10111	01100	11001	
00100	10001	01010	11111	
01000	11101	00110	10011	
10000	00101	11110	01011	
11000	01101	01110	00011	
10010	00111	11100	01001	

Properties of standard array decoding

What about Maximum Likelihood Decoding (nearest neigbour) strategy ?

- Standard array decoding is in accordance with MLD as,

$$\boxed{d(\mathbf{l}_i + \mathbf{c}_j, \mathbf{c}_j) \leqslant d((\mathbf{l}_i + \mathbf{c}_j, \mathbf{c_h}) \ \ \forall \mathbf{c}_j}$$

– This means that if $\mathbf{r} = \mathbf{l}_i + \mathbf{c}_j$ is received then $\mathbf{c}_j$ is closest to $\mathbf{r}$ than any other codeword (see Lemma 3.8).

- Two cases to consider:
  1. $\mathbf{l}_i$ is unique vector of least weight in C_i - then $\mathbf{c}_j$ closest to $\mathbf{l}_i + \mathbf{c}_j$, OK.
  2. $\mathbf{l}_i$ not unique (more than one vector of least weight) still $\mathbf{c}_j$ closest to r than any other $\mathbf{c}_h$

Properties of standard array decoding II

Theorem Let C with $w(C) = d$. If $\mathbf{x}$ is such that

$$w(\mathbf{x}) \leqslant \left\lfloor \frac{d-1}{2} \right\rfloor$$

then $\mathbf{x}$ **is unique element of minimum weight** in its coset and thus a coset leader.

Proof Suppose $w(\mathbf{x}) \leqslant \lfloor \frac{d-1}{2} \rfloor$ and there is $\mathbf{y} : w(\mathbf{y}) \leqslant w(\mathbf{x})$. Since $\mathbf{x} - \mathbf{y} \in C$ (there are in the same coset) we have,

$$w(\mathbf{x} - \mathbf{y}) \leqslant w(\mathbf{x}) + w(\mathbf{y}) \leqslant \lfloor \frac{d-1}{2} \rfloor + \lfloor \frac{d-1}{2} \rfloor \leqslant d - 1$$

Contradicts the fact $w(C) = d$, unless $\mathbf{x} = \mathbf{y}$.

Standard array decoding - algorithm

Standard array decoding for linear codes

Precomputation: Construct a standard array S

Let $\mathbf{r}$ be a received vector

1. Find $\mathbf{r}$ in the standard array S
2. Correct $\mathbf{r}$ to the codeword at the top of its column

S will correct any e or fewer errors but also of weight $e + 1$ if the pattern appears as a coset leader.

Standard array decoding - example

Assume in previous example of a $(5,2)$ code that $\mathbf{r} = (10111)$

coset leaders

00000	10101	01110	11011	codewords
00001	10100	01111	11010	
00010	10111	01100	11001	
00100	10001	01010	11111	
01000	11101	00110	10011	
10000	00101	11110	01011	
11000	01101	01110	00011	
10010	00111	11100	01001	

Syndrome decoding

Few problems with standard array decoding:

- storing a standard array (e.g. $q = 2, n = 40$)
- locating the received vector in the table (cannot sort it)

More efficient approach is called syndrome decoding

- The syndrome of $\mathbf{x}$ is computed as $H\mathbf{x}^T$. Why we do that ?
- It turns out that all the elements in the same coset of C have the same syndrome !

Theorem Two vectors $\mathbf{x}$, $\mathbf{y}$ are in the same coset of C **if and only if** they have the same syndrome, i.e. $H\mathbf{x}^T = H\mathbf{y}^T$.

Syndrome decoding - algorithm

Proof(sketch) $\mathbf{x}, \mathbf{y} \in C_k \Rightarrow \mathbf{x} = \mathbf{l}_k + \mathbf{c}_i; \mathbf{y} = \mathbf{l}_k + \mathbf{c}_j$. Then,

$$H\mathbf{x}^T = H(\mathbf{l}_k + \mathbf{c}_i)^T = H\mathbf{l}_k^T = H\mathbf{y}^T$$

- The main idea is to establish 1-1 correspondence between coset leaders and syndromes

Syndrome decoding for linear codes

Precomputation: 1-1 one correspondence between coset leaders and syndromes

Let $\mathbf{r}$ be a received vector and H the parity check

1. Compute the syndrome $s = H\mathbf{r}^T$ of $\mathbf{r}$
2. Find the coset leader $\mathbf{l}$ associated with $\mathbf{s}$
3. Correct $\mathbf{r}$ to $\mathbf{r} - \mathbf{l}$

Syndrome decoding - example

We follow the same example our $(5, 2)$ code C, $\mathbf{r} = 10111$ with,

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}; H = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}; \mathbf{s} = H\mathbf{r}^T = 010$$

coset leaders				syndrome
00000	10101	01110	11011	000
00001	10100	01111	11010	001
00010	10111	01100	11001	010
00100	10001	01010	11111	100
01000	11101	00110	10011	011
10000	00101	11110	01011	101
11000	01101	01110	00011	110
10010	00111	11100	01001	111

Not needed !

Standard array vs. syndrome decoding

- Suppose C is a binary $(70, 50)$ code, then $|C| = 2^{50}$ codewords.
- The number of cosets is $2^{70}/2^{50} = 2^{20}$.

– Comparing the two strategies,

	Standard array	Syndrome
Storage	2^{70}	$2^{20}(70 + 20)$
Dec. Computation	Search 2^{70} entries	Search 2^{20} entries

- But we can further improve the decoding storage.
- Only keep correspondence between syndromes and weights of coset leaders !

Step-by-step decoding

For our previous example we would have,

Syndrome	Weight of coset leaders
000	0
001	1
010	1
100	1
011	1
101	1
110	2
111	2

The algorithm processes $\mathbf{r}$ by flipping one bit at a time, and checks whether the vector is moved to a lighter coset leader.

Step-by-step decoding

Step-by-step decoding for linear codes II

Precomputation: Set up 1-1 one correspondence between syndromes and weights coset leaders
Let $\mathbf{r}$ be a received vector and H the parity check

1. Set $i = 1$
2. Compute $H\mathbf{r}^T$ and the weight w of corresponding coset leader
3. If $w = 0$, stop with $\mathbf{r}$ as the transmitted codeword
4. If $H(\mathbf{r} + \mathbf{e}_i)^T$ has smaller associated weight than $H\mathbf{r}^T$, set $\mathbf{r} = \mathbf{r} + \mathbf{e}_i$.
5. Set $i = i + 1$ and go to 2.

Read example 27 in the textbook for further understanding how the algorithm works.

Weight distribution - motivation

- Weight distribution gives a detailed description of the number of codewords of certain weight in a code.
- For a (non)linear $[n, M]$ code C let

$$A_i = \#\{\mathbf{c} : w(c) = i, \mathbf{c} \in C\}$$

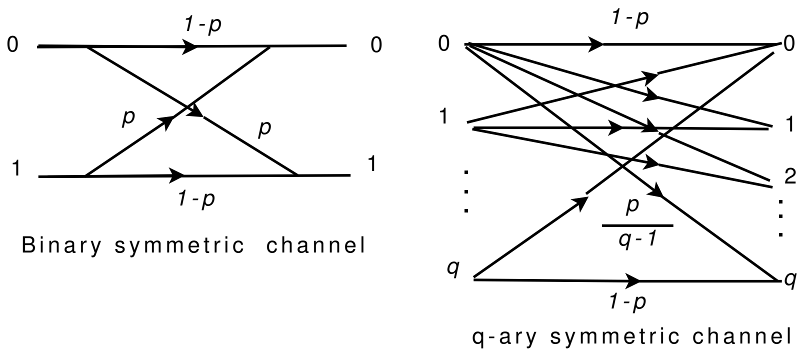
- Vector $(A_0, A_1, \ldots, A_n)$ is called the weight distribution of C.

Two main reasons for studying the weight distribution:

– For determining the probability of incorrectly decoded received vectors
– For deriving Mac-Williams identity

q-ary symmetric channel

Assumption is that any symbol from $\mathcal{A}$ has the same probability of being transformed into another symbol.



The probability of error

Assumption: C is an (n, k)-code over $F = GF(q)$ and the zero codeword is sent.

The probability that some (specified) codeword of weight i is received is,

$$\left(\frac{p}{q-1}\right)^i (1-p)^{n-i}, 0 \leqslant i \leqslant n$$

- Of interest is to compute the probability that an error goes undetected(codeword goes into another codeword)

$$\sum_{i=1}^{n} A_i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i}$$

NOTE: Correct the formulae in the textbook (there summation goes from $i = 0$)

The probability of error II

- If C has distance $d = 2t + 1$ and incomplete decoding(only decode if $d(\mathbf{r}, \mathbf{c}) \leqslant t$) is used then

$$Pb(\textbf{correct decoding}) = \sum_{i=1}^{t} \binom{n}{i} p^i (1-p)^{n-1}$$

– What is the probability if both correction and detection are used ?

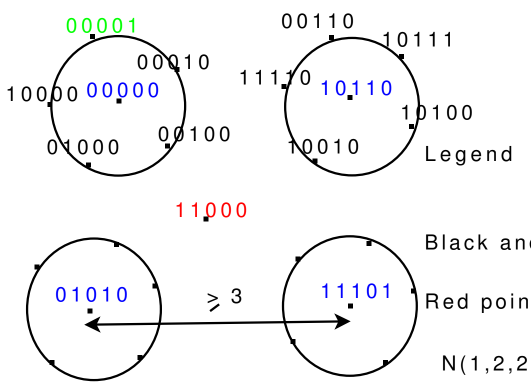
  - Define $N(i, h, s)$ as follows:

– No codewords of weight i then $N(i, h, s) = 0$, otherwise

$$N(i, h, s) = \#\{\mathbf{x} : w(\mathbf{x}) = h \ \& \ d(\mathbf{x}, \mathbf{c}) = s \text{ for fixed } \mathbf{c} : w(\mathbf{c}) = i\}.$$

– $N(i, h, s)$ independent of the given codeword of weight i (exercise 98)

Error probability and codeword spheres



Legend

Black and blue correctly decoded

Red points incorrectly decoded

N(1,2,2)= 0 for c=(00001)

N(1,2,1)=4 { (00011),(00101),(01001), (10001)}

d=2e+1=3; e=1

Spheres of radius 1

The probability of decoding error

The number of vectors of weight h at distance s of the codewords of weight i is

$$A_i N(i, h, s)$$

- To get improperly decoded vector it must lie in a sphere of another codeword of radius t other than that which was sent.
- The probability of receiving a particular vector of weight h is,

$$\left(\frac{p}{q-1}\right)^h (1-p)^{n-h}$$

- What does the following expression then relate to ?

$$A_i N(i, h, s) \left(\frac{p}{q-1}\right)^h (1-p)^{n-h}$$

The probability of decoding error II

So if zero codeword is sent the probability of decoding it as some codeword of weight i is,

$$\sum_{h=0}^{n} \sum_{s=0}^{t} A_i N(i, h, s) \left(\frac{p}{q-1}\right)^h (1-p)^{n-h}$$

- If $i \geqslant 1$ then a decoding error has occurred. Thus the probability of a decoding error is,

$$\sum_{i=1}^{n} \sum_{h=0}^{n} \sum_{s=0}^{t} A_i N(i, h, s) \left(\frac{p}{q-1}\right)^h (1-p)^{n-h}$$

- Again to compute this probability - need weight distribution !

Weight enumerators

Small codes - list the codewords and find weight distribution. E.g.

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Then $C = \{0000, 1100, 0011, 1111\}$ thus $A_0 = 1, A_2 = 2, A_4 = 1$.

> For linear codes we can find out the weight distribution of a code given the weight distribution of its dual (or vice versa)

Definition Let C be an (n, k)-code over F with weight distribution $(A_0, A_1, \ldots, A_n)$. The weight enumerator of C is defined as,

$$W_C(x, y) = \sum_{i=0}^{n} A_i x^{n-i} y^i.$$

Weight enumerators II

- For each $\mathbf{u} \in V_n(F)$ we define $P(\mathbf{u}) = x^{n-w(\mathbf{u})} y^{w(\mathbf{u})}$. Then,

$$\sum_{\mathbf{u} \in C} P(\mathbf{u}) = \sum_{i=0}^{n} A_i x^{n-i} y^i = W_C(x, y)$$

Example For $C = \{0000, 1100, 0011, 1111\}$ we can compute

$$P(0000) = x^4; P(0011) = x^2y^2; P(1100) = x^2y^2; P(1111) = y^4$$

- This formalism is proved useful for deriving MacWilliams identity
- Identity is valid for any linear code and if e.g. dual code of C is of small dimension we get its weight distribution and then obtain weight distribution of C

MacWilliams identity - preparation (optional)

Only consider $q = 2$. Easily generalized to $A = GF(p^k)$.

- Define a function,

$$g_n(\mathbf{u}) = \sum_{\mathbf{v} \in V_n} (-1)^{\mathbf{u}\cdot\mathbf{v}} P(\mathbf{v}), \ \mathbf{u}, \mathbf{v} \in V_n(GF(2))$$

Lemma 3.11 If C is a binary (n, k)-code then

$$\sum_{\mathbf{u} \in C^\perp} P(\mathbf{u}) = \frac{1}{|C|} \sum_{\mathbf{u} \in C} g_n(\mathbf{u}).$$

Proof (sketch) Write

$$\sum_{\mathbf{u} \in C} g_n(\mathbf{u}) = \sum_{\mathbf{u} \in C} \sum_{\mathbf{v} \in V_n} (-1)^{\mathbf{u}\cdot\mathbf{v}} P(\mathbf{v}) = \sum_{\mathbf{v} \in V_n} P(\mathbf{v}) \sum_{\mathbf{u} \in C} (-1)^{\mathbf{u}\cdot\mathbf{v}}$$

MacWilliams identity - preparation II (optional)

Proof (cont.) Easy to verify that,

$$\sum_{\mathbf{u} \in C} (-1)^{\mathbf{u}\cdot\mathbf{v}} = \begin{cases} |C| & \text{if } \mathbf{v} \in C^\perp \\ 0 & \text{if } \mathbf{v} \notin C^\perp \end{cases}$$

Therefore,

$$\sum_{\mathbf{u} \in C} g_n(\mathbf{u}) = |C| \sum_{\mathbf{v} \in C^\perp} P(\mathbf{v}).$$

The following result is also needed (Lemma 3.12 in the textbook),

$$g_n(\mathbf{u}) = (x + y)^{n - w(\mathbf{u})} (x - y)^{w(\mathbf{u})}.$$

Proved by induction on n !

MacWilliams identity

Theorem If C is a binary (n, k) code with dual $C^{\perp}$ then,

$$W_{C^{\perp}}(x, y) = \frac{1}{2^k} W_C(x + y, x - y).$$

Proof Let the weight distribution of C be $(A_0, A_1, \ldots, A_n)$. Then,

$$\begin{aligned}
\sum_{\mathbf{u} \in C^{\perp}} P(\mathbf{u}) &= \frac{1}{|C|} \sum_{\mathbf{u} \in C} g_n(\mathbf{u}) \text{ Lemma 3.11} \\
&= \frac{1}{|C|} \sum_{\mathbf{u} \in C} (x + y)^{n - w(\mathbf{u})} (x - y)^{w(\mathbf{u})} \text{ Lemma 3.12} \\
&= \frac{1}{|C|} \sum_{i=0}^{n} A_i (x + y)^{n-i} (x - y)^i = \frac{1}{|C|} W_C(x + y, x - y)
\end{aligned}$$

MacWilliams identity - example

Assume given is a $(6, 3)$ binary code C with (Ex. 10)

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

The weight distribution of C is $(1, 0, 0, 4, 3, 0, 0)$. What is the weight distribution of $C^{\perp}$?

$$\begin{aligned}
W_C(x + y, x - y) &= (x + y)^6 + 4(x + y)^3(x - y)^3 + 3(x + y)^2(x - y)^4 \\
&= \ldots = 8x^6 + 32x^3y^3 + 24x^2y^4
\end{aligned}$$

Then, by MacWilliams identity,

$$W_{C^{\perp}}(x, y) = \frac{1}{8} W_C(x+y, x-y) = x^6 + 4x^3y^3 + 3x^2y^4 = W_C(x, y)$$

Computing the weight distribution

Assume we have a linear $(70, 50)$ code C.

- Cannot compute the probability of incorrect decoding - need the weight distribution of C.
- But the dual code is a $(70, 20)$ code and from $G \to H \to C^{\perp} = span(H)$ we can compute the weight distribution of $C^{\perp}$.
- MacWilliams gives us the weight distribution of C.
- The main question is how to construct good linear codes (apart from Hamming codes)
- E.g. the code used in Mariner was a Reed-Muller $(32, 6)$ code of min. distance 16 !

Conclusions

- Many nice algebraic properties for linear codes (not always the case for nonlinear codes)
- Connection to dual code
- General decoding strategies: standard array and syndrome decoding
- Further decoding optimizations possible

Chapter 5

Coding theory - Constructing New Codes

Contents of the chapter:

- Constructing new codes
- Basic methods for constructions
- Some bounds
- Other construction methods
- Elias codes

Hamming codes and perfect codes - reminder

- Introduced Hamming codes as example of perfect codes
- Perfect codes : Spheres around codewords exhaust the whole space
- Hamming (binary) code has parameters

$$(n = 2^r - 1, 2^r - 1 - r, 3) \quad r \geqslant 3$$

Syndrome decoding - reminder

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix};$$

coset leaders				*syndrome*
00000	10101	01110	11011	000
00001	10100	01111	11010	001
00010	10111	01100	11001	010
00100	10001	01010	11111	100
01000	11101	00110	10011	011
10000	00101	11110	01011	101
11000	01101	01110	00011	110
10010	00111	11100	01001	111

Array not needed !

MacWilliams identity-reminder

Theorem

If C is a binary (n,k) code with dual $C^{\perp}$ then,

$$W_{C^{\perp}}(x,y) = \frac{1}{2^k} W_C(x+y, x-y).$$

$$W_C(x,y) = \sum_{i=0}^{n} A_i x^{n-i} y^i.$$

A_i weight distribution - number of codewords of weight i.

Introduction

- So far we looked at Hamming codes
- These codes are only defined for some specific lengths, have certain minimum distance and dimension.

Can we get other codes out of the known ones ?

YES, by using the techniques of puncturing, extending, taking crossover sections ...

Detecting more errors

- Assume we want to detect 3 errors
- Hamming (7,4,3) code cannot be used - 2 errors can be detected
- Can we construct a new code from (7,4,3) code that detects 3 errors ?
- YES, slightly worse rate 4/8 instead of 4/7, but possible.

Simple extension - example

Take the Hamming (7,4,3)- $\mathbf{0} \in \mathbb{F}_2^7$ and 7 cyclic shifts of (1101000)

$$\text{8 words} \left\{ \begin{array}{c} 0000000 \\ 1101000 \\ 0110100 \\ \vdots \\ 1010001 \end{array} \right. \qquad \text{8 complements} \left\{ \begin{array}{c} 1111111 \\ 0010111 \\ 1001011 \\ \vdots \\ 0101110 \end{array} \right.$$

Add to these codewords one coordinate (extending) as,

$$\mathbf{c}_{i,8} = \oplus_{j=1}^{7} \mathbf{c}_{i,j}$$

E.g. (1101000) $\rightarrow$ (11010001), we get (8,4) code $\overline{H}$

Extending codes

Definition

If C is a code of length n over $\mathbb{F}_q$ the extended code $\overline{C}$ is defined by,

$$\overline{C} := \{(c_1, \ldots, c_n, c_{n+1}) | (c_1, \ldots, c_n) \in C, \sum_{i=1}^{n+1} c_i = 0\}$$

- Note that the extended code is linear if C is linear (exercise)
- From the Hamming (7,4,3) code we get an (8,4,4) code, i.e. $n + 1 \leftarrow n$ and $d + 1 \leftarrow d$! Always possible ?
- How is $\overline{C}$ specified in terms of generator and parity check matrix ?

Minimum distance of extended codes

– Note that in case of (7,4,3) code we are forced to have both even and odd weight codewords :

- If only even weight then $d(C) \geqslant 4$ for a (7,4,3) code C.
- We cannot have only odd weight codewords as adding 2 codewords of odd weight gives a codeword of even weight (exercise)
- Finally note that for odd weight the parity (extended bit) is 1 - all together we get an (8,4,4) code.

Question : Why we cannot proceed and get $(n + i, k, d + i)$?

Another view of the problem

Assume we can do that : What is the consequence on relative distance,

$$\delta = \frac{d}{n}.$$

We would have,

$$\delta = \frac{d+i}{n+i} \to 1 \qquad i \to \infty.$$

Clearly not possible for arbitrary k and n.

Some intuition

We "know" there is no (9,4,5) code (at least cannot extend (8,4,4) to get this one)

- Maybe the space was to small.
- But then we can find 16 codewords if the length $n = 10$, i.e. $(10, 4, 5)$ code
- Seems logical ain't it ?
- Visit `http://www.codetables.de/` to find out the answer.

Decomposing generator matrix

- Assume C is a binary (n, k) linear code with min. distance d.
- Generator matrix G is an $k \times n$ binary matrix
- IDEA: Split G into 2 parts (decompose) and check whether you can get required d

Decomposing generator matrix - example

Example

Let us consider the existence of (9,4,5) code

$$G = \left[\begin{array}{ccccc|ccccc} 1 & 1 & \dots & 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ & & G_1 & & & & & G_2 & & \end{array} \right]$$

G_1 is a $(k-1) \times d$ and G_2 is a $(k-1) \times (n-d)$ binary matrix.

Let d' denote the min. distance of the code generated by G_2

To each codeword of C_2 there correspond 2 codewords of C

At least one of these codewords has weight $\leqslant \frac{1}{2}d$ on the first d position. Therefore $d' \geqslant \frac{1}{2}d$ (finish at home for $d = 5$)

Generator matrix of the extended code

If C is a linear code with generator matrix G then,

$$\overline{G} = \left[\; G \quad G_{n+1} + \sum_{i=1}^{n} G_i \quad = 0 \right]$$

where G_i denotes the i-th column of G.

For instance the generator matrix of the (7,4,3) code is,

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} ; \sum_i G_i = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Parity check matrix of the extended code

If H is a parity check of C then the parity check of $\overline{C}$ is,

$$\overline{H} := \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ & & & & 0 \\ & & H & & 0 \\ & & & & \vdots \\ & & & & 0 \end{bmatrix}$$

Check that $\overline{\mathbf{c}}\overline{H}^T = 0$ or $\overline{H}\overline{\mathbf{c}}^T = 0$ for all $\overline{\mathbf{c}} \in \overline{C}$!

If C has an odd minimum distance d then $\overline{C}$ has minimum distance $d+1$ (all weights and distances even).

Augmenting the code

Simply adding more codewords to the original code.

The most common way is to add $\mathbf{1}$ to the generator matrix (if $\mathbf{1}$ is not already in the code)

$$G^{(a)} = \begin{bmatrix} G \\ \mathbf{1} \end{bmatrix}$$

- Alternatively, for a binary (n, k, d) code C the augmented code is,

$$C^{(a)} = C \cup \{\mathbf{1} + C\}$$

What are the general properties of the augmented code ?

Augmenting the code II

Adding the codewords has the following effect:

- The length n is the same
- Number of codewords (dimension of the code) increases
- Minimum distance decreases in general,

$$d^{(a)} = \min\{d, n - d'\}$$

where d' is the largest weight of any codeword in C

$$\text{8 words} \left\{ \begin{array}{c} 0000000 \\ 1101000 \\ 0110100 \\ \vdots \\ 1010001 \end{array} \right. \quad \text{8 complements} \left\{ \begin{array}{c} 1111111 \\ 0010111 \\ 1001011 \\ \vdots \\ 0101110 \end{array} \right.$$

Expurgating the code

Definition

Expurgation: Throwing away the codewords of the code.

CAUTION : It can turn a linear code into a nonlinear one. E.g. throwing away 5 out of 16 codewords of a (7,4,3) code results in a nonlinear code.

The most common way is to throw away codewords of odd weight if the number of odd and even weight codewords is equal.

For which codes we have the above situation ?

Expurgating the codewords of odd weight

Facts

If C is a binary (n, k, d) code containing words of both odd and even weight then (exercise)

$$|\{\mathbf{c} \in C : wt(\mathbf{c} = \ \ odd\ \}| = |\{\mathbf{c} \in C : wt(\mathbf{c} = \ \ even\ \}| = 2^{k-1}$$

Almost always the case (but not exactly) !

Expurgating the code - example

We throw away the odd weight codewords of a (6,3,3) code generated by,

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}; C = \begin{cases} 000000 \\ 100111 \\ 001101 \\ 101010 \\ 011011 \\ 111100 \\ 010110 \\ 110001 \end{cases}$$

The minimum distance of the new (6,2) code is $d = 4$, i.e. we get a (6,2,4) code !

Puncturing the code

Puncturing : Inverse process to extension

- Delete one coordinate of the code (suitable) to get C^*.

Example

From a (3,2,2) code by puncturing we get a (2,2,1) code,

$$(3,2,2) \text{ code } \begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix} \qquad (2,2,1) \text{ code } \begin{matrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{matrix}$$

Deleting the coordinate has the following effect:

- The length n drops by 1
- Number of codewords (dimension of the code) unchanged
- Minimum distance drops by one (in general)

Shortening the code by taking a cross-section

The operation of throwing out the codewords that all have the same value in one coordinate and deleting the coordinate position.

For simplicity, say we shorten $c_1 = 0$ *in the example below.*

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \rightarrow G' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

- From the original code we have thrown out all the codewords that start with one, i.e. $c_1 = 1$.
- Shortening can be seen as expurgating followed by puncturing.

Shortening as expurgating + puncturing

In the previous example we would delete the codewords,

$$C = \begin{cases} 0000 \\ 0011 \\ 0110 \\ 0101 \\ 1100 \\ 1111 \\ 1010 \\ 1001 \end{cases} \quad C' = \begin{cases} 000 \\ 011 \\ 110 \\ 101 \end{cases}$$

This is a linear $(3, 2, 2)$ code !

Shortening the code making it nonlinear

What if we keep the codewords that have $c_1 = 1$. In the previous example we would delete the codewords,

$$C = \begin{cases} 0000 \\ 0011 \\ 0110 \\ 0101 \\ 1100 \\ 1111 \\ 1010 \\ 1001 \end{cases} \quad C' = \begin{cases} 100 \\ 111 \\ 010 \\ 001 \end{cases}$$

This is a nonlinear $[3, 4, 1]$ code !

Lengthening the code

Inverse operation to shortening.

Can be viewed as:

1. extending (adding new columns to generator matrix)
2. followed by augmenting (adding rows to the extended generator matrix)

$$G = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \rightarrow G^L = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Summary of basic construction methods

Defining the "redundancy" as $r = n - k$ for a binary (n, k) code

- Augmenting: Fix n; increase k; decrease r.
- Expurgating: Fix n; decrease k; increase r.
- Extending: Fix k; increase n; increase r.
- Puncturing: Fix k; decrease n; decrease r.
- Lengthening: Fix r; increase n; increase k.
- Shortening: Fix r; decrease n; decrease k.

Apart from these there are several other techniques to construct new codes from the old ones.

Good and the best codes

For a given alphabet of q symbols and length of the code we can try to maximize:

- Number of the codewords given a designed minimum distance (might be a linear or nonlinear code)
- Maximize a dimension of a linear code k for a given minimum distance (or vice versa)

Even for small parameters hard to find good codes

Good and the best codes -example

Example

A rather complicated construction from the 60's gave a $[10, 38, 4]$ code - a good code.

Until 1978 it was believed this was the best possible code for $n = 10, d = 4$.

But then $[10, 40, 4]$ was found - the BEST CODE.

Strange language

Example

- Using “strange” language over binary alphabet: 30 letters and 10 decimal digits
- Can use $[10, 40, 4]$ and correct a single error; detect 3
- Problem implementation : coding, decoding ?
- What about linear codes : only $k = 5$ - 32 codewords for $n = 10$, need $n = 11$
- Wasting the bandbred

Existence of codes - example

Example

We can ask a question: Is there a binary (5,3,3) linear code ?

ANSWER: Prove it by hand (no need for decomposition)!

- Need to construct a binary 3×5 generator matrix G so that any linear combination has weight at least 3 !

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ ? & ? & ? & ? & ? \end{bmatrix} \text{ try e.g. } G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & ? & ? \end{bmatrix} \text{ NO}$$

Any nonzero combination of 3 rows yields some $\mathbf{c}$ s.t. $wt(\mathbf{c}) < 3$.

Finding the best codes - example II

So we know there is no (5,3,3) binary code, easy to construct (5,2,3) code, e.g.

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Can we have more than 4 codewords in a nonlinear code ?

Turns out that we cannot do better, though the **upper and lower bounds** say that $4 \leqslant M \leqslant 6$.

Thus, out of 16 codewords of $wt \geqslant 3$ we cannot find 5 codewords s.t. their mutual distance is $\geqslant 3$!

Lower and upper bounds on codes

Useful measures - know the range of possible :

- number of codewords for given n, d
- upper bound on dimension (for linear codes)
- also lower bound on dimension (number of codewords)

Singleton bound - introduction

Motivation

:

– there is an $(n, n, 1)$ binary code - $G = I_n$

– there is an $(n, n-1, 2)$ binary code - $G = [I_{n-1}\mathbf{1}]$

Can we generalize these results to a $(n, n-d+1, d)$ code ?

Or why should we only ask for $k \leqslant n-d+1$ and not better!

The Singleton bound shows that this is indeed the best possible, over any alphabet.

Singleton bound

Theorem

(Singleton) If C is an (n, k, d) code then $d \leqslant n - k + 1$.

Proof.

Use projection of the codewords to the first $(k-1)$ coordinates

- 2^k codewords $\Rightarrow \exists \mathbf{c}_1, \mathbf{c}_2$ having the same first $k-1$ values
- Then $d(\mathbf{c}_1, \mathbf{c}_2) \leqslant n - (k-1) = n - k + 1$ thus $d \leqslant n - k + 1$

□

Singleton bound -example

Example

For instance, we cannot have (7,4,5) code but can we construct (7,4,4) code ?

NO, the codes having $d = n - k + 1$ exists for some special n, k, q

Quite loose upper bound !

Singleton bound - generalization

Generalization

- Assume we have an $[n, M, d]$ code over $\mathbb{F}_q$ and punctures it $d-1$ times.
- The punctured code is an $[n-d+1, M, 1]$ code, i.e. the M punctured words are different (can we have $d > 1$?).
- Thus,

$$M \leqslant q^{n-d+1}$$

The bound is quite lose.

MDS and perfect codes

Codes that meet the Singleton bound, i.e., satisfy $k = n - d + 1$, are called Maximum Distance Separable (MDS) codes.

Perfect Codes codes meet the Hamming bound - e.g. the Hamming codes and two codes discovered by Golay.

Facts

MDS codes and perfect codes are incomparable:

- there exist perfect codes that are not MDS and
- there exist MDS codes that are not perfect.

Each meets an incomparable optimality criterion. The most famous class of MDS codes is the Reed-Solomon codes.

Hamming bound

Sometimes called *sphere packing bound* - generalization of a sphere-packing condition,

$$|C| \underbrace{\sum_{i=0}^{e} \binom{n}{i} (q-1)^i}_{V_q(n,e)} = q^n \quad \text{perfect code} \quad d = 2e+1.$$

Theorem

(Hamming bound) If $q, n, e \in \mathbb{N}$, $d = 2e+1$ then,

$$|C| \leqslant q^n / V_q(n, e); \quad .$$

Proof: The spheres $B_e(\mathbf{c})$ are disjoint.

Hamming bound - applications

Could construct $(n, n-i, i+1)$ codes for $i = 0, 1$

For $n = 7$ Singleton bound says - no (7,4,5) code.

It says nothing about (7,4,4) code !

Example

For $n = 7$ the Hamming bound gives,

$$|C| \leqslant 2^7 / (1+7) = 16$$

Thus the Hamming $(7, 4, 3)$ code meets the upper bound !

Therefore, no (7,4,4) code !!

Hamming bound - example

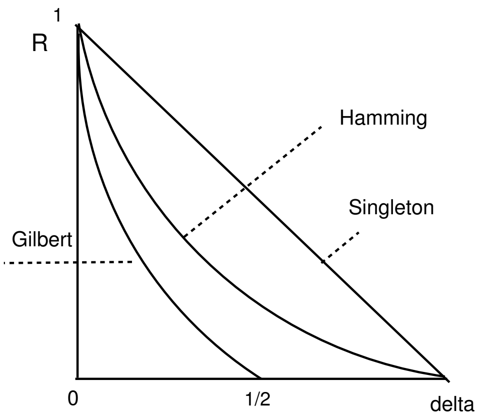
Example

- Another example is the UB on M for a $[5, M, 3]$ code
- Applying the Hamming bound we get,

$$|C| = M \leqslant \frac{2^5}{6} = 5.3 = 5.$$

Note that Singleton bound (generalized) gives $M \leqslant 2^{n-d+1} = 8$.

Hamming vs Singleton bound



Upper bounds - binary alphabet

Other construction methods

Apart from already mentioned methods :

- Need something called ($\mathbf{u}$,$\mathbf{u}$ + $\mathbf{v}$) construction
- And direct product method

– Why do we need them ?

Commonly used in construction of good codes, and the latter comes close to Shannon (in the easiest way)

($\mathbf{u}$,$\mathbf{u}$ + $\mathbf{v}$) construction

In general, let C_i be a binary $[n, M_i, d_i]$ code $(i = 1, 2)$ and define,

$$C : \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) | \mathbf{u} \in C_1, \mathbf{v} \in C_2\}$$

Example

Take 2 codes given by

$$G_1 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad G_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Our codewords would be

$$\begin{matrix} (1011 & 1100) \\ (1011 & 0011) \\ & \vdots \end{matrix}$$

The length of the code is easy $2n$!

What about the dimension and minimum distance ?

$(\mathbf{u},\mathbf{u}+\mathbf{v})$ construction - properties

Theorem

Then C is a $[2n, M_1M_2, d]$ code, where $d := \min\{2d_1, d_2\}$

Proof.

Consider $(\mathbf{u}_1, \mathbf{u}_1+\mathbf{v}_1)$ and $(\mathbf{u}_2, \mathbf{u}_2+\mathbf{v}_2)$.

1. If $\mathbf{v}_1 = \mathbf{v}_2$ and $\mathbf{u}_1 \neq \mathbf{u}_2$ then $d \geqslant 2d_1$
2. If $\mathbf{v}_1 \neq \mathbf{v}_2$ the distance is (triangle ineq.)

$$wt(\mathbf{u}_1-\mathbf{u}_2)+wt(\mathbf{u}_1-\mathbf{u}_2+\mathbf{v}_1-\mathbf{v}_2) \geqslant wt(\mathbf{v}_1-\mathbf{v}_2) \geqslant d_2$$

□

$(\mathbf{u},\mathbf{u}+\mathbf{v})$ construction - example

An abstract justification.

Example

Take for C_2 an $[8, 20, 3]$ obtained by puncturing a $[9, 20, 4]$ code

$\boxed{\text{What code to use for } C_1 \text{ with respect to } d \text{ and M ?}}$

Take an (8,7) even weight code as C_1 - to increase M !

The construction gives a $[16, 20 \cdot 2^7, 3]$ - at present no better code is known for $n = 16, d = 3$.

Direct product codes - Motivation

- The method known as the direct product of codes

Applications include:

– getting good codes

– proving the nonexistence of certain linear codes

– deriving a class of asymptotically good codes

Direct product codes - motivation

- Main idea: Collaboration of two (or more) codes
- Efficiently used in the compact disc application to combat the burst errors.
- More errors then expected can be corrected (more on this in the last lecture)

Even more important - asymptotically good codes were constructed by P. Elias in 1954.

One of a few construction that approaches Shannon

Direct product codes

Notation:

– A and B are binary linear (n_A, k_A, d_A) and (n_B, k_B, d_B) codes, respectively

– $\mathcal{R}$ a set of all binary $n_A \times n_B$ matrices over $GF(2)$ - vector space of dimension $n_B \times n_A$

Example

One basis for the vector space of 2×2 matrices is:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Direct product codes - definition

Definition

The direct product $A \otimes B$ is the code consisting of all

$$n_A \times n_B$$

matrices with the property that:

– each matrix column is a codeword of A and

– each matrix row is a codeword of B.

Kronecker product - linear algebra

Direct product codes - example

Example

$$G_A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \; A = \begin{cases} 000 \\ 101 \\ 011 \\ 110 \end{cases} ; \; G_B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \; B = \begin{cases} 0000 \\ 1100 \\ 0011 \\ 1111 \end{cases}$$

Then for instance the 3×4 matrix $M \in \mathcal{R}$ where,

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \text{or} \quad M^{(2)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

2^{12} matrices only $16 = 2^{k_A k_B}$ of these satisfy definition. The **codewords** corresponding to M and $M^{(2)}$ are:

$$\mathbf{c} = (111111000011) \;\; \mathbf{c}^{(2)} = (111100111100)$$

Direct product code - property

Facts

The product code is “clearly” linear :

– all zero matrix $\in \mathcal{R}$

– as A and B are linear then any linear combination of the columns(rows) of $C = A \otimes B$ is a valid column(row)

What about the minimum distance - i.e. minimum weight of the codewords?

Direct product code - min. distance

We can show that $d_C \geqslant d_A d_B$

Justification

If $A \neq 0$ then there is a nonzero row with weight $\geqslant d_B$.

Then A has at least d_B nonzero columns of weight $\geqslant d_A$ so

$$wt(A) \geqslant d_A d_B.$$

One can show that $\mathbf{a}^T\mathbf{b} \in \mathcal{R}$ for $\mathbf{a} \in A,\ \mathbf{b} \in B$.

If $wt(\mathbf{a}) = d_A$ and $wt(\mathbf{b}) = d_B$ then $wt(\mathbf{a}^T\mathbf{b}) = d_A d_B$

Recall $\mathbf{c} = (111111000011)$ $\mathbf{c}^{(2)} = (111100111100)$

Direct product code - property II

Example

Let ${\mathbf{g}_1^A}^T = (101)^T$ and $\mathbf{g}_1^B = (1100)$. Then

$${\mathbf{g}_1^A}^T \mathbf{g}_1^B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot [1\ 1\ 0\ 0] = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Corresponding codeword is $\mathbf{c} = (1100|0000|1100)$

The basis of the code $C = A \otimes B$ is the set,

$${\mathbf{g}_i^A}^T \mathbf{g}_j^B \quad 1 \leqslant i \leqslant k_A; 1 \leqslant j \leqslant k_B$$

$\mathbf{g}_i^A$ and $\mathbf{g}_j^B$ are the rows of the generator matrices G_A and G_B.

Iterative approach

To summarize :

$\boxed{\text{Therefore, } C = A \otimes B \text{ is a linear } (n_A n_b, k_A k_B, d_A d_B) \text{ code.}}$

Iterative product - a sequence of direct product codes,

$$C = A^{(1)} \otimes A^{(2)} \otimes \cdots A^{(r)} \otimes \ldots$$

– Idea used by P. Elias utilizing the **extended Hamming codes** - simplest approach to get closer to Shannon's bound - codes when $n \to \infty$

Remark : The part on the construction of Elias codes is optional - for interested students

Elias codes - preliminaries

Start with extended Hamming codes $C_1(\mathcal{H}^{2^m})$ and $C_2(\mathcal{H}^{2^{m+1}})$ of respective length $n_1 = 2^m$ and $n_2 = 2^{m+1}$

Assumption: Codes used on a BSC with bit error probability p, and $n_1 p < 1/2$.

Definition

- Define: $V_1 := C_1$ and $V_2 = C_1 \otimes C_2$
- Let V_i be an (n_i, k_i) code
- Let E_i be the expected number of errors per block **after decoding**.

  Continue in this way:
- If V_i has been defined then $V_{i+1} = V_i \otimes \mathcal{H}^{2^{m+i}}$

Properties of recursion

Facts

From the definition of recursion we have:

$$\begin{aligned} n_{i+1} &= n_i 2^{m+i} \\ k_{i+1} &= k_i(2^{m+i} - m - i - 1) \end{aligned}$$

For extended Hamming codes we know that (Example 3.3.4 J. H. von Lint):

$$E_{i+1} \leqslant E_i^2 \quad \text{and} \quad E_1 \leqslant (n_1 p)^2 \leqslant 1/4$$

Thus, these codes have the property $E_i \to 0$ when $i \to \infty$.

Can we express n_i in terms of m and i ?

Some math - arithmetic sum

The sum of first 5 integers is $1 + 2 + 3 + 4 + 5 = \frac{5 \cdot 6}{2}$.

Recursion

$$\begin{aligned} i = 1 \qquad & n_1 = 2^m \\ i = 2 \qquad & n_2^{V_2} = 2^m \cdot 2^{m+1} = 2^{2m+1} \\ i = 3 \qquad & n_3^{V_3} = 2^{2m+1} \cdot 2^{m+2} = 2^{3m+1+2} \\ & \vdots \\ & n_i = 2^{mi+(1+2+\ldots+i-1)} = 2^{mi+\frac{1}{2}i(i-1)} \end{aligned}$$

$$n_i = 2^{mi+\frac{1}{2}i(i-1)}; \qquad k_i = n_i \prod_{j=0}^{i-1} \left(1 - \frac{m+j+1}{2^{m+j}}\right)$$

Comments on Elias codes

If $R_i = k_i/n_i$ denotes the rate of V_i then,

$$R_i \to \prod_{j=0}^{i-1}\left(1 - \frac{m+j+1}{2^{m+j}}\right) > 0 \quad \text{for} \quad i \to \infty$$

The Elias sequence of codes has the following properties:

- Length $n \to \infty$ but $R_i \not\to 0$!
- At the same time $E_i \to 0$
- Elias codes have $d_i = 4^i$ so $\frac{d_i}{n_i} \to 0$ for $i \to \infty$.

One of a few systematic construction that accomplishes the Shannon's result !

Conclusions

- Constructing good (best) codes is commonly a serious combinatorial challenge
- Methods mentioned in this lecture do not exhaust the possibilities (of course)
- Many open problems
- Elias codes comprise the basic goal in coding theory : possibility of errorless transmission over a noisy channel
- Of course, the problem of efficient decoding of long codes is important

Chapter 6

Coding theory - Bounds on Codes

Contents of the chapter:

- Shannons theorem revisited
- Lower bounds
- Upper bounds
- Reed-Muller codes

Linear versus nonlinear codes

- Previous lecture: Several good codes (and a few best) was mentioned
- How can we claim these were good codes ?
- Ideally, the number of codewords meets the upper bound.
- Very rare situations, even for small n - recall $[5, 4, 3]$ code, UB=5.
- In this case # codewords same for linear and nonlinear code

Linear versus nonlinear codes II

Example: the nonexistence of $(12, 5, 5)$ code and can find a $[12, 32, 5]$ code

Keep in mind - linear codes used in practice (easy encoding and decoding); nonlinear codes - "combinatorial challenges"

Apart from encoding "strange" languages :)

Main goals of coding theory

– $\mathcal{A}$ is the alphabet of q symbols; $k = \log_q |C|$

- Given k, d, q find an $[n, k, d]_q$ code that minimizes n
- Given n, d, q find an $[n, k, d]_q$ code that maximizes k
- Given n, k, q find an $[n, k, d]_q$ code that maximizes d
- Given n, k, d find an $[n, k, d]_q$ code that minimizes q

The last one is not obvious, but empirically good codes are obtained by reducing q

- Rate of the code $R = \frac{k}{n}$
- Relative distance $\delta = \frac{d}{n}$

Some families of codes

Some families of codes (binary) :

- Hamming codes: $(2^r - 1, k = n - r, d = 3)$ - good rate but small distance
- Hadamard codes: $[n, 2n, n/2]$ - good distance but (very) small rate
- Reed-Muller codes: $(2^r, r + 1, 2^{r-1})$-good distance but (very) small rate

Need asymptotically good (family) of codes in the Shannon's sense - fixed rate $P_E \to 0$

Optimal codes

Definition

$$A(n, d) := \max\{M : \text{ an } [n, M, d] \text{ code exists }\}$$

A code C such that $|C| = A(n, d)$ is called optimal

- Good codes are long (Shannon) - given p of a BSC we can have $P_e \to 0$, $R > 0$, when $n \to \infty$
- Number of errors in a received word is np; $\Rightarrow d$ must grow at least as fast as $2np$ to correct errors ($d = 2e + 1$)

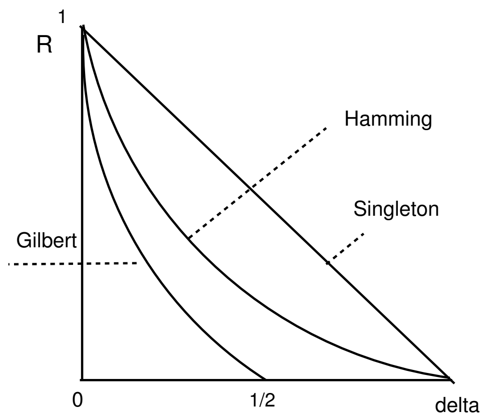
Optimal codes II

– Given the rate R we ask how large $\delta = d/n$ is (as a function of n)

Useful notation:

– $V_q(n, r) := |B_r(\mathbf{c})| = \sum_{i=0}^{r} \binom{n}{i}(q-1)^i$ - # of sphere of radius r

Hamming vs Singleton bound - reminder



Upper and lower bounds - binary alphabet

Gilbert - Varshamov bound

Almost trivial but powerful lower bound

There is an asymptotic version of the bound concerning $n \to \infty$

- Until 1982 it was believed that $R(\delta)$ equals this bound
- Bound was improved for $q \geqslant 49$ using methods of algebraic geometry (tedious proof)
- Using Shimura curves (modular curves) to construct sequence of Goppa codes beating the GV bound for $\alpha(\delta)$

– Maximal code - An $[n, M, d]$ code which is not contained in any $[n, M+1, d]$ code

Gilbert - Varshamov bound II

Theorem

(GV bound) For $n, d \in \mathbb{N}$, $d \leqslant n$, we have,

$$A(n,d) \geqslant q^n / V_q(n, d-1).$$

Proof.

- Let the $[n, M, d]$ code C be maximal, i.e. there is no word in $\mathcal{A}^n$ with distance $\geqslant d$ to all words in C
- That is, the spheres $B_{d-1}(\mathbf{c})$, $\mathbf{c} \in C$ cover $\mathcal{A}^n$
- Means - the sum of their volumes, $|C| V_q(n, d-1)$ exceeds q^n

$\square$

Constructing good long codes

- In the previous lecture we took some codes (extended Hamming) and constructed $C = C_1 \otimes C_2 \otimes \cdots$
- Length $n \to \infty$ but $R_i \not\to 0$!
- At the same time $E_i \to 0$
- These codes have $d_i = 4^i$ so $\frac{d_i}{n_i} \to 0$ for $i \to \infty$.
- Method required iteration and usage of direct product codes (but efficient).

GV as a construction method

Interpretation of the GV bound :

– Start with any $\mathbf{c} \in \mathcal{A}^n$ and update $\mathcal{A}^n \leftarrow \mathcal{A}^n \setminus B_{d-1}(\mathbf{c})$

– Take a new codeword from $\mathcal{A}^n$ and update

– Proceed as long as there are vectors in $\mathcal{A}^n$ until the code is maximal

- Method is constructive but there is no structure in the code.
- Gives an exponential time deterministic algorithm and nonlinear codes

Gilbert - Varshamov bound for linear codes

Theorem

(GV bound LC) If $n, d, k \in \mathbb{N}$ satisfy

$$V_q(n, d-1) < q^{n-k+1}$$

then an (n, k, d) code exists.

Proof.

- Let C_{k-1} be an $(n, k-1, d)$ code. Since,

$$|C_{k-1}|V_q(n, d-1) = q^{k-1}V_q(n, d-1) < q^n,$$

  C_{k-1} is not maximal, i.e. $\exists \mathbf{x} \in \mathcal{A}^n : d(\mathbf{x}, \mathbf{c}) \geqslant d, \ \forall \mathbf{c} \in C_{k-1}$

- The code spanned by C_{k-1} and $\mathbf{x}$ is an (n, k, d) (exercise)

$\square$

GV bound - example II

GV bound for LC is sufficient but not necessary. E.g. we may want to deduce if there exists a binary $(15, 7, 5)$ code.

Check the GV condition, $n = 15, \; k = 7, \; d = 5$.

$$V_q(n, d-1) < q^{n-k+1} \Leftrightarrow \sum_{i=0}^{4} \binom{n}{i} \not< 2^9.$$

Clearly, not satisfied - so the GV bound **does not tell us whether such a code exists !**

- There is a linear BCH (cyclic) code with such parameters !

Varshamov construction - linear codes

A randomized polynomial time construction.

Algorithm:

1. Pick a $k \times n$ matrix G at random
2. Let $C = \{\mathbf{x}G | \mathbf{x} \in \{0, 1\}^k\}$

- Claim: With high probability C has 2^k distinct elements.
- Furthermore, their pairwise distance is at least d provided that
$$2^k - 1 < 2^n / V_2(n, d-1).$$

Few words on probability

Let us consider binary vectors of length 4.

Probability of randomly selecting vector of weight 1 is

$$Pb(wt(\mathbf{c} = 1)) = \frac{4}{16}$$

What is the probability of selecting another vector of weight 1

$$Pb(wt(\mathbf{c} = 1)) = \frac{3}{15} < \frac{4}{16}$$

Conclusion: we may say,

$$Pb(2 \text{ vectors of wt } 1) < 2Pb(1 \text{ vector of wt } 1\ )$$

Varshamov construction - proof

Proof.

1. Suffices to show that for every non-zero $\mathbf{x}$,

$$\mathbf{x}G \notin B(\mathbf{0}, d-1)$$

2. Fix $\mathbf{x}$. Then $\mathbf{x}G$ is a random vector. It falls in $B(\mathbf{0}, d-1)$ with prob. $V_2(n, d-1)/2^n$.

3. By union bound ($Pb(\cup_{i=1}^n A_i) \leqslant \sum_{i=1}^n Pb(A_i)$), the probability that there is $\mathbf{x}$ such that $\mathbf{x}G \in V_2(n, d-1)$ is at most

$$(2^k - 1)V_2(n, d-1)/2^n$$

4. If this quantity is less than 1 then such a code exists. If this prob. $\ll 1$ then we find such a code with higher prob..

□

Is GV bound tight ?

Previous construction claims that random codes approaches GV bound (asymptotically).

Are we done ? NO, we cannot check it (long codes)

Dominating belief in coding community: GV bound is tight ! Not exactly true, many counterexamples:

– Hamming codes beat GV

– Cyclic codes beat GV

Is GV bound tight - example

Example

Hamming codes specified by $n = 2^r - 1$, $k = 2^r - 1 - r$, $d = 3$

Need to compute

$$V_2(n, d-1) = \sum_{i=0}^{2} \binom{2^r - 1}{i}$$

and to compare with $2^{n-k+1} = 2^{r+1}$.

$$\text{for } \; r = 3 \qquad \sum_{i=0}^{2} \binom{2^r - 1}{i} = 1 + 7 + \frac{7 \cdot 6}{2} = 29 \not< 16$$

GV bound not satisfied **but there exists the Hamming code**.

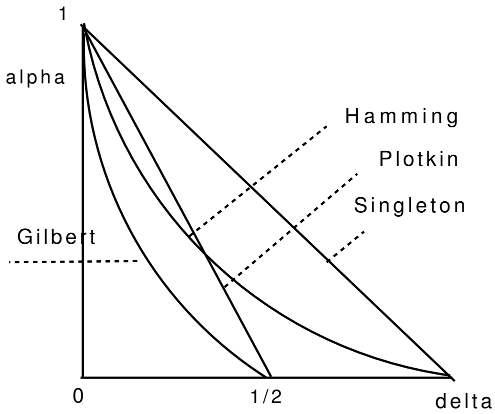
Other upper bounds

We are trying to come close to GV bound from above - squeezing the space in between

Many upper bounds:

- Have seen Singleton and Hamming
- There are Plotkin, Griesmer and plenty of more sophisticated bounds
- Elias, McEliece et al., Johnson
- Linear Programming bounds ("state-of-the-art" bounds)

Plotkin vs Hamming and Singleton bound



Upper bounds - binary alphabet

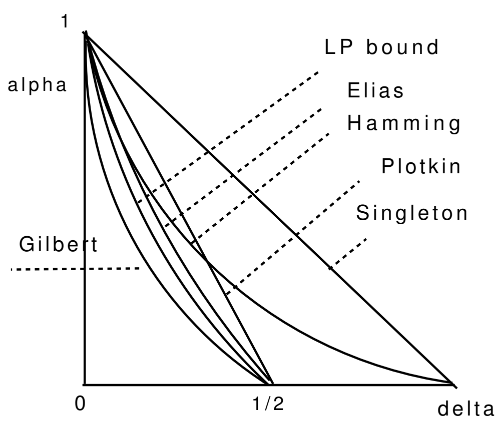
Is LP bound tight ?

Example

Back to the same example, we had :

- $A(13, 5) \leqslant 512$ - Singleton bound
- Claimed that there was a nonlinear [13,64,5] code (construction)
- LP bound gives $A(13, 5) \leqslant 64$
- Means that $A(13, 5) = 64$!

Linear programming bound - asymptotic



Reed-Muller codes

- Large class of codes
- Not the same minimum distance for any length as Hamming codes
- Recall that Mariner used RM code, (32,6,16) code
- Closely related to nonlinearity of Boolean functions

Hamming codes - reminder

- Recall that the Hamming code was defined as a single-error correcting code with parameters $n = (q^r - 1)/(q - 1)$ and $k = n - r$.
- The parity check of a Hamming code is an $r \times n$ matrix.
- For a binary Hamming code $q = 2$ so that $n = 2^r - 1$, $k = 2^r - 1 - r$.
- Its parity check consists of all nonzero vectors of length r (over GF(2)).

Introducing Reed-Muller codes II

- Let us define,

$$B_r = [H_r, \mathbf{0}] = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_r \end{bmatrix}$$

- The size of B is $r \times 2^r$.

Notation: $\mathbf{1}$ is the row vector of all-ones of length 2^r.

Reed-Muller code

Definition

The first order Reed-Muller code denoted $R(1, r)$ is the subspace generated by the vectors $\mathbf{1}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$.

- Obviously the length of the code is 2^r, what are the dimension and min. distance ?

Note that the generator matrix of $R(1, r)$ is,

$$G = \left[\frac{\mathbf{1}}{B_r}\right] = \left[\frac{\mathbf{1}}{H_r \mathbf{0}}\right]$$

Theorem

$R(1, r)$ is a $(2^r, r+1, 2^{r-1})$ code.

Parameters of 1-st order Reed-Muller code

Proof.

Need to prove the results on dimension and min. distance

Dimension: can find identity $r \times r$ as a submatrix of B_r and $\mathbf{1}$ is clearly independent from other rows. Thus, $k = r + 1$.

To show that $d = 2^{r-1}$ one proves that $w(c) = 2^{r-1}$ for any $c \in C \setminus \{\mathbf{1}, \mathbf{0}\}$ □

– The main idea of the proof (see textbook) is to use:
- the fact that $H_r\mathbf{0}$ has all distinct r-tuples as its columns;

Parameters of 1-st order RM code

Example

$$H_3\mathbf{0} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \text{ all vectors of} GF(2)^3$$

Can $\mathbf{c} = (11000000)$ be in the code ?

No, easily checked. Can $\mathbf{c} = (01100000)$ be in the code ?

No, ...

But if $\mathbf{c}$ is in the code then it must be of weight 4.

RM versus Hamming codes

We switched dimensions approximately :

	ReedMuller	*Hamming*
dimension	$r+1$	$2^r - r - 1$
length	2^r	$2^r - 1$
d	2^{r-1}	3

Reed-Muller code example

Let us construct a first-order Reed-Muller code $R(1,3)$ for $r = 3$.

$$H_3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \text{ all nonzero vectors of} GF(2)^3$$

$$B_3 = H_3\mathbf{0} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \text{ all vectors of} GF(2)^3$$

$$G = \left[\frac{\mathbf{1}}{H_3\mathbf{0}}\right] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Chapter 7

Reed-Muller codes

Contents of the chapter:

- Direct product of RM
- Decoding RM
- Hadamard transform

Reed-Muller code example

Let us construct a first-order Reed-Muller code $R(1,3)$ for $r = 3$.

$$H_3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \text{ all nonzero vectors of} GF(2)^3$$

$$B_3 = H_3\mathbf{0} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \text{ all vectors of} GF(2)^3$$

$$G = \left[\frac{\mathbf{1}}{H_3\mathbf{0}}\right] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Reed-Muller code(reminder)

Definition

The first order Reed-Muller code denoted $R(1,r)$ is the subspace generated by the vectors $\mathbf{1}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$.

- Obviously the length of the code is 2^r, what are the dimension and min. distance ?

Note that the generator matrix of $R(1,r)$ is,

$$G = \left[\frac{\mathbf{1}}{B_r}\right] = \left[\frac{\mathbf{1}}{H_r\mathbf{0}}\right]$$

Theorem

$R(1,r)$ is a $(2^r, r+1, 2^{r-1})$ code.

Using Reed-Muller codes in direct product

- Direct product of 2 codes A and B was defined using basis vectors of 2 codes,

$$g_A^T g_B$$

- If A is an (n_1, k_1, d_1) and B is an (n_2, k_2, d_2) code then $C = A \otimes B$ is a $(n_1 n_2, k_1 k_2, d_1 d_2)$ code (easy to remember).

Example

Want to construct a (16,9) linear code !

Can we use Reed-Muller codes of the form $(2^r, r+1, 2^{r-1})$?

Using Reed-Muller codes in direct product II

Example

Fits perfect for $r = 2$ we get a (4,3,2) RM code

Then using two such codes (same) in direct product we get a linear code,

$$(n_1 n_2, k_1, k_2, d_1 d_2) = (16, 9, 4)$$

- You may say, so what ?
- Go to our favorite website `www.codetables.de` and check for $n = 16,\ k = 9$
- No better linear codes than (16,9,4) code for fixed $n = 16$ and $k = 9$!

Construction example

Example

A (4,3,2) RM code C is easily constructed using,

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

- Encoding is simple, e.g. $\mathbf{m} = (011)$

$$\mathbf{m}G = (011) \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} = (1100)$$

Construction example II

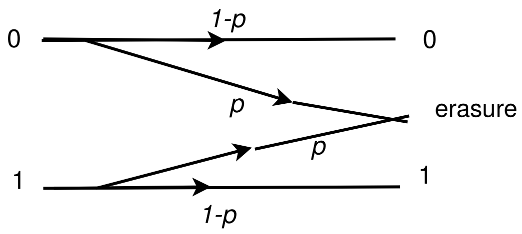
Example

What are the codewords of a big (16,9,4) code $V = C \otimes C$? For instance $\mathbf{c}_1 = (0110)$ and $\mathbf{c}_2 = (0101)$ gives,

$$\mathbf{c}_1^T \mathbf{c}_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} (0101) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- Cannot correct 2 errors with such a code but can correct 3 erasures !

Erasure channel



Binary erasure channel

- The receiver knows where are the possible errors !
- Later we show we can always correct $d-1$ erasures.

Decoding erasures

Example

Given is the received word with 3 erasures for a (16,9,4) code,

$$\mathbf{r} = \begin{pmatrix} 0 & E & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & E & 0 & E \end{pmatrix}$$

Decoding strategy:

- Correct erasures in each column using the erasure correction for a (4,3,2) RM code.
- Correct erasures in each row using the erasure correction for a (4,3,2) RM code.

Decoding erasures -example

Example

Correcting erasures in columns gives

$$\mathbf{r} = \begin{pmatrix} 0 & E & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & E & 0 & 0 \end{pmatrix}$$

Cannot correct 2nd column but now correct rows:

$$\mathbf{r} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Decoding Reed-Muller codes

We know how to construct an R-M code but can we efficiently decode ?

Need to introduce few concepts:

- Proper ordering
- Hadamard matrices
- Hadamard transform

The decoding process turns out to be quite simple - just a matrix vector multiplication !

Hadamard matrix

Definition

A Hadamard matrix H_n is an $n \times n$ matrix with integer entries $+1$ and -1 whose rows are pairwise orthogonal as real numbers.

Example

The matrix

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

is a Hadamard matrix.

Hadamard matrix - history

Dates from the mid of 19-th century. Many applications:

- Combinatorial theory
- Quantum cryptography (complex Hadamard matrices), Boolean functions
- Measuring the spectrum of light etc.

Hadamard conjecture

Facts

Hadamard conjectured that such a matrix of size $4k \times 4k$ could be constructed for any k !

Hard to disprove as the smallest order for which no construction is known is 668!

Easy case $4k = 2^u$ - Sylvester construction.

$$\begin{bmatrix} H & H \\ H & -H^T \end{bmatrix}$$

Properties of Hadamard matrices

Hadamard matrices of order 1,2, and 4 are,

$$H_1 = [1] \quad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

- Equivalent definition: $n \times n$ ± 1 matrix such that,

$$H_n H_n^T = nI_n.$$

Properties of Hadamard matrices II

Example

$$H_2 H_2^T = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

H_n has the following properties,

1. $H_n^T = nH_n^{-1}$ thus $H_n^T H_n = nI_n$ - columns orthognal as well
2. Changing rows (columns) again Hadamard matrix
3. Multiplying rows(columns) by (-1) again Hadamard matrix

Sylvester construction

Hadamard matrices of order 2^r easily constructed recursively,

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$$

Consider again H_2 and H_4,

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

- Useful in decoding R-M code; recall that the length of an R-M code is 2^r

Proper ordering of binary vectors

Simply binary representation of integers with the leftmost bit as the least significant bit. For example,

$$12 = 0011 = 0 \cdot 2^0 + 0 \cdot 2^1 + 2^2 + 2^3$$

Formally the proper ordering P_r of binary r-tuples defined recursively for $1 \leqslant i \leqslant r-1$,

$$\begin{aligned} P_1 &= [0, 1] \\ \text{if } P_i &= [\mathbf{b}_1, \dots, \mathbf{b}_{2^i}] \text{ then } P_{i+1} = [\mathbf{b}_1 0, \dots, \mathbf{b}_{2^i} 0, \mathbf{b}_1 1, \dots, \mathbf{b}_{2^i} 1] \end{aligned}$$

Proper ordering - example

Example

- Binary triples would be ordered as,

$$P_3 = [000, 100, 010, 110, 001, 101, 011, 111]$$

Take $n = 2^r$ and $\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{n-1} \in GF(2)^r$ in proper order

- Construct $H = [h_{ij}]; 0 \leqslant i, j \leqslant n-1$ where

$$h_{ij} = (-1)^{\mathbf{u} \cdot \mathbf{v}} \quad \text{" $\cdot$ "} - \text{ dot product}$$

Hadamard matrix - an alternative view

Example

Let $r = 2$. Then,

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

E.g. $h_{12} = (-1)^{(100)\cdot(010)} = (-1)^{1\cdot 0+0\cdot 1+0\cdot 0} = -1^0 = 1$

Introducing Hadamard transform

Example

Consider $r = 3$ and $\mathbf{r} = (11011100)$.

Any single $\mathbf{u}$ of length r picks up a component of $\mathbf{r}$.

$\mathbf{r}(\underbrace{110}_{\mathbf{u}}) = 1$ picks up the 4-th component of $\mathbf{r}$

$\mathbf{r}(000) = 1$ takes 1-st component etc. (counting from 1)

Then **define**

$$\mathbf{R}((110)) = (-1)^{\mathbf{r}(110)} = (-1)^1 = -1.$$

Continuing $\mathbf{R} = (-1, -1, 1, -1, -1, -1, 1, 1)$

Introducing Hadamard transform II

Important tool for encoding/decoding, studying Boolean functions etc.

IDEA: From a binary r-tuple obtain a real scalar $\mathbf{R}(\mathbf{u})$ using,

$$\mathbf{u} \in \mathbb{F}_2^{\mathbf{r}}, \quad \mathbf{r} \in \mathbb{F}_2^{2^{\mathbf{r}}} \quad \rightarrow \quad \mathbf{r}(\mathbf{u}) \rightarrow \quad \mathbf{R}(\mathbf{u}) = (-\mathbf{1})^{\mathbf{r}(\mathbf{u})}$$

Alternatively, the mapping $\mathbf{r} \rightarrow \mathbf{R}$ is defined as

$$0 \mapsto 1 \qquad\qquad 1 \mapsto -1$$

Hadamard transform

Definition

The Hadamard transform of the 2^r-tuple $\mathbf{R}$ is the 2^r-tuple $\hat{\mathbf{R}}$ where for any $\mathbf{u} \in \mathbb{F}_2^r$

$$\hat{\mathbf{R}}(\mathbf{u}) = \sum_{\mathbf{v} \in \mathbb{F}_2^r} (-1)^{\mathbf{u}\cdot\mathbf{v}} \mathbf{R}(\mathbf{v}).$$

Using $\mathbf{R}(\mathbf{v}) = (-\mathbf{1})^{\mathbf{r}(\mathbf{v})}$ we get,

$$\hat{\mathbf{R}}(\mathbf{u}) = \sum_{\mathbf{v} \in \mathbb{F}_2^r} (-1)^{\mathbf{u}\cdot\mathbf{v}+\mathbf{r}(\mathbf{v})}.$$

Essentially we measure the distance to linear (Boolean) functions !

Hadamard transform - example

Example

Given $\mathbf{r} = (11011100)$ we want to compute $\hat{\mathbf{R}}(110)$, i.e. $\mathbf{u} = (110)$

$$\begin{aligned}
\hat{\mathbf{R}}(110) &= \sum_{\mathbf{v}\in\mathbb{F}_2^3} (-1)^{(110)\cdot\mathbf{v}+\mathbf{r}(\mathbf{v})} \\
&= (-1)^{(110)\cdot(100)+\mathbf{r}(100)} + (-1)^{(110)\cdot(010)+\mathbf{r}(010)} \\
&= (-1)^{(110)\cdot(001)+\mathbf{r}(001)} + (-1)^{(110)\cdot(110)+\mathbf{r}(110)} + \cdots \\
&= (-1)^{1+1} + (-1)^{1+1} + (-1)^{0+0} + (-1)^{0+1} + \cdots = 6
\end{aligned}$$

- Need to compute 7 more values for other vectors $\mathbf{u}$
- Alternatively, $\hat{\mathbf{R}}$ can be defined as (exercise 25), $\hat{\mathbf{R}} = \mathbf{R}H$, where H is a Hadamard matrix of order 2^r !

Computing Hadamard transform - example

Example

For $\mathbf{r} = (11011100)$ we have computed $\mathbf{R} = (-1, -1, 1, -1, -1, -1, 1, 1)$. Then,

$$\mathbf{R}H_8 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \\ -6 \\ -2 \\ -2 \\ 2 \\ 2 \\ -2 \end{bmatrix}^T$$

We do not get 6 as in the previous example but -2. Proper ordering is important !

Computing Hadamard transform - explanation

The first computation was performed using,

$$B_3 = H_3\mathbf{0} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \text{ all vectors of} GF(2)^3$$

Decoding 1-st order RM codes

Ordered columns of B_r are used to associate the distinct binary r-tuples with the coordinate positions in $\mathbf{r}, \mathbf{R}, \hat{\mathbf{R}}$.

Theorem

(Main theorem) $\hat{\mathbf{R}}(\mathbf{u})$ is the number of 0's minus the number of 1's in the binary vector,

$$\mathbf{t} = \mathbf{r} + \sum_{i=1}^{r} u_i \mathbf{v}_i$$

where $\mathbf{u} = (u_1, u_2, \ldots, u_r)^T$ and $\mathbf{v}_i$ is the i-th row of B_r.

Developing the decoding procedure

- The previous result allows us to include the Hadamard values for measuring the minimum distance. The number of 0's in

  $\mathbf{t} = \mathbf{r} + \sum_{i=1}^{r} u_i \mathbf{v}_i$ is,

$$t_0 = 2^r - w(\mathbf{t}) = 2^r - w(\mathbf{r} + \sum_{i=1}^{r} u_i \mathbf{v}_i) = 2^r - d(\mathbf{r}, \sum_{i=1}^{r} u_i \mathbf{v}_i)$$

- Then obviously $t_1 = d(\mathbf{r}, \sum_{i=1}^{r} u_i \mathbf{v}_i)$.

- Now using $\hat{\mathbf{R}}(\mathbf{u}) = t_0 - t_1$ we have,

$$\hat{\mathbf{R}}(\mathbf{u}) = 2^r - 2d(\mathbf{r}, \sum_{i=1}^{r} u_i \mathbf{v}_i)$$

Developing the decoding procedure II

1. Another way to compute t_0 is,

$$t_0 = w(\mathbf{1} + \mathbf{t}) = w(\mathbf{1} + \mathbf{r} + \sum_{i=1}^{r} u_i \mathbf{v}_i) = d(\mathbf{r}, \mathbf{1} + \sum_{i=1}^{r} u_i \mathbf{v}_i)$$

2. Then $t_1 = 2^r - d(\mathbf{r}, \mathbf{1} + \sum_{i=1}^{r} u_i \mathbf{v}_i)$ so that

$$\hat{\mathbf{R}}(\mathbf{u}) = 2d(\mathbf{r}, \mathbf{1} + \sum_{i=1}^{r} u_i \mathbf{v}_i) - 2^r$$

3. Finally,

$$\overbrace{d(\mathbf{r}, \sum_{i=1}^{r} u_i \mathbf{v}_i) = \frac{1}{2}(2^r - \hat{\mathbf{R}}(\mathbf{u})) \quad ; d(\mathbf{r}, \mathbf{1} + \sum_{i=1}^{r} u_i \mathbf{v}_i) = \frac{1}{2}(2^r + \hat{\mathbf{R}}(\mathbf{u}))}^{\text{decoding formulas}}$$

Deriving the decoding procedure

Suppose $\mathbf{r} \in \mathbb{F}_2^{2^r}$ is a received vector. Our goal is to decode $\mathbf{r}$ to the codeword closest to $\mathbf{r}$.

Facts

- For any binary r-tuple $\mathbf{u} = (u_1, \ldots, u_r)$, $\mathbf{u}B_r = \sum_{i=1}^r u_i \mathbf{v}_i$.
- An $(r+1)$ message tuple can be viewed as $\mathbf{m} = (u_0, \mathbf{u})$ where $u_0 \in \{0,1\}$ and $\mathbf{u} \in \mathbb{F}_2^r$

The transmitted codeword is,

$$\mathbf{c} = \mathbf{m}G = (u_0, \mathbf{u}) \begin{bmatrix} \mathbf{1} \\ B_r \end{bmatrix} = u_0 \cdot \mathbf{1} + \sum_{i=1}^r u_i \mathbf{v}_i.$$

Connection to encoding

Decoding formulas considers two cases $u_0 = 0$ and $u_0 = 1$!

$$\boxed{\text{Finding } \mathbf{c} \text{ closest to } \mathbf{r} = \text{find } \mathbf{c} \text{ minimizing } d(\mathbf{r}, \mathbf{c})}$$

Exactly what is given by decoding formulas,

$$d(\mathbf{r}, \sum_{i=1}^r u_i \mathbf{v}_i) = \frac{1}{2}(2^r - \hat{\mathbf{R}}(\mathbf{u})) \quad ; d(\mathbf{r}, \mathbf{1} + \sum_{i=1}^r u_i \mathbf{v}_i) = \frac{1}{2}(2^r + \hat{\mathbf{R}}(\mathbf{u})),$$

for $\mathbf{c} = \sum_{i=1}^r u_i \mathbf{v}_i$ and $\mathbf{c} = \mathbf{1} + \sum_{i=1}^r u_i \mathbf{v}_i$!

Computing $\hat{\mathbf{R}}(\mathbf{u})$ for all $\mathbf{u} \in \mathbb{F}_2^r \Leftrightarrow d_{\mathbf{c} \in C}(\mathbf{r}, \mathbf{c})$

Decoding RM codes

From the decoding formula $d_{\mathbf{c}\in C}(\mathbf{r},\mathbf{c})$ is minimized by $\mathbf{u}$ which minimizes,

$$\min_{\mathbf{u}}\{2^r - \hat{\mathbf{R}}(\mathbf{u}), 2^r + \hat{\mathbf{R}}(\mathbf{u})\}$$

- We are looking for $\mathbf{u}$ that maximizes the magnitude of $\hat{\mathbf{R}}(\mathbf{u})$!
- Assuming we have found a unique $\mathbf{u}$ maximizing $|\hat{\mathbf{R}}(\mathbf{u}))|$, we have 2 cases,

$$\mathbf{c} = \begin{cases} \sum_{i=1}^{r} u_i\mathbf{v}_i; & \hat{\mathbf{R}}(\mathbf{u})) > 0, u_0 = 0 \\ \mathbf{1} + \sum_{i=1}^{r} u_i\mathbf{v}_i; & \hat{\mathbf{R}}(\mathbf{u})) < 0, u_0 = 1 \end{cases}$$

Decoding algorithm for RM$(1, r)$

INPUT: $\mathbf{r}$ a received binary vector of length 2^r; B_r with columns in the proper ordering P_r; H a Hadamard matrix $H = H(2^r)$.

1. Compute $\mathbf{R} = (-1)^{\mathbf{r}}$ and $\hat{\mathbf{R}} = \mathbf{R}H$
2. Find a component $\hat{\mathbf{R}}(\mathbf{u})$ of $\hat{\mathbf{R}}$ with max. magnitude, let $\mathbf{u} = (u_1, \ldots, u_r)^T$
3. If $\hat{\mathbf{R}}(\mathbf{u}) > 0$, then decode $\mathbf{r}$ as $\sum_{i=1}^{r} u_i\mathbf{v}_i$
4. If $\hat{\mathbf{R}}(\mathbf{u}) < 0$, then decode $\mathbf{r}$ as $\mathbf{1} + \sum_{i=1}^{r} u_i\mathbf{v}_i$

Decoding RM$(1, r)$ - example

Example Let B_3 for a RM$(1,3)$ be in proper order,

$$B_3 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} \text{ all vectors of} GF(2)^3$$

The corresponding generator matrix is,

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

For a received $\mathbf{r} = (01110110)$ need to compute $\mathbf{R}$ and $\hat{\mathbf{R}}$!

Direct product of RM Decoding RM Hadamard transform

Decoding RM$(1, r)$ - example cont.

Easy to compute $\mathbf{R} = (-1)^{\mathbf{r}} = (1, -1, -1, -1, 1, -1, -1, 1)$. Also,

$$\hat{\mathbf{R}} = \mathbf{R}H = (-2, 2, 2, 6, -2, 2, 2, -2).$$

Thus, $|\hat{\mathbf{R}}(\mathbf{u})| = 6$ and $\mathbf{u} = (110)^T$.

Since $\hat{\mathbf{R}}(110) = 6 > 0$ then,

$$\mathbf{c} = \sum_{i=1}^{3} u_i \mathbf{v}_i = 1 \cdot \mathbf{v}_1 + 1 \cdot \mathbf{v}_2 + 0 \cdot \mathbf{v}_3 = (011000110)$$

Given $\hat{\mathbf{R}}$ and decoding formula we can find distance to each of the codewords. E.g. $\hat{\mathbf{R}}(000) = -2$ so $d(\mathbf{r}, \mathbf{0}) = 5$ and $d(\mathbf{r}, \mathbf{1}) = 3$

Chapter 8

Fast decoding of RM codes and higher order RM codes

Contents of the chapter:

- Fast Hadamard transform
- RM codes and Boolean functions
- Self-Dual codes

Decoding complexity

Recall that the main decoding step is to compute,

$$\hat{\mathbf{R}} = \mathbf{R}H$$

Complexity of decoding = computing the vector matrix product.

Example

Mariner was using RM(1,5) code to correct up to 7 errors. Received vectors of length $2^5 = 32$ are multiplied with $H(2^5)$ requiring c.a. $2^{2r+1} = 2^{11}$ operations (back in the 70's).

What if r is large, can we do better ?

Fast Hadamard transform RM codes and Boolean functions Self-Dual codes

Decoding complexity II

- YES, we utilize the structure of the Hadamard matrix to reduce the complexity to $r2^r$.

Definition

For $A = [a_{ij}]$ and $B = [b_{ij}]$ of order m and n respectively define the *Kronecker product* as $mn \times mn$ matrix,

$$A \times B = [a_{ij}B]$$

So we are back again to product codes.

Decoding erasures revisited

- Our (16,9,4) code could only correct 1 error !
- Assume we could correct this error using RM decoding. Then number of computations is:

$$16 \times (16 + 15) \approx 16 \times 32 = 512 = 2^9$$

where blue denotes multiplications and green additions.

- Using our decoding approach we only need to check rows (or columns) which gives,

$$\text{Nmb. of operations } = 4 \times (4 \times (4 + 3)) = 112$$

The Kronecker product

The Hadamard matrix $H(2^r)$ can be viewed as,

$$H(2^r) = \underbrace{H_2 \times H_2 \times \cdots \times H_2}_{r \text{ times}}$$

Example

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

A useful property of the Kronecker product is (Lemma 4.3),

$$(A \times B)(C \times D) = AC \times BD$$

The fast Hadamard transform

IDEA Split the computation into chunks (called butterfly structure)

Theorem

For a positive integer r,

$$H(2^r) = M_{2^r}^{(1)} M_{2^r}^{(2)} \cdots M_{2^r}^{(r)},$$

where $M_{2^r}^{(i)} = I_{2^{r-i}} \times H_2 \times I_{2^{i-1}}$.

It turns out that less operations are required for computing $\mathbf{R}M_{2^r}^{(1)} M_{2^r}^{(2)} \cdots M_{2^r}^{(r)}$ then $\mathbf{R}H(2^r)$ directly !

Decomposition - example

For $r = 2$ we need to show that,

$$H_4 = M_4^{(1)} M_4^{(2)}$$

where,

$$M_4^{(1)} = I_2 \times H_2 \times I_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes [1] =$$

$$I_2 \times H_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Decomposition - example II

$$M_4^{(2)} = H_2 \times I_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Finally we confirm below that $M_4^{(1)} \times M_4^{(2)} = H_4$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Computing M matrices

$$M_8^{(1)} = I_4 \times H_2 \times I_1 = \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix} \cdots$$

$$M_8^{(3)} = H_2 \times I_4 = \begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & 1 \\ 1 & & & & -1 & & & \\ & 1 & & & & -1 & & \\ & & 1 & & & & -1 & \\ & & & 1 & & & & -1 \end{bmatrix}$$

Computing fast Hadamard transform - example

In the previous example we had
$\mathbf{R} = (-1)^{\mathbf{r}} = (1, -1, -1, -1, 1, -1, -1, 1)$ and

$$\hat{\mathbf{R}} = \mathbf{R}H(2^3) = (-2, 2, 2, 6, -2, 2, 2, -2).$$

Computing via M matrices gives,

$$\begin{aligned} \mathbf{R}M_8^{(1)} &= (0, 2, -2, 0, 0, 2, 0, -2) \\ (\mathbf{R}M_8^{(1)})(M_8^{(2)}) &= (-2, 2, 2, 2, 0, 0, 0, 4) \\ (\mathbf{R}M_8^{(1)}M_8^{(2)})M_8^{(3)} &= (-2, 2, 2, 6, -2, 2, 2, -2) \end{aligned}$$

- Many zeros in M matrices yields more efficient computation.

Comparing the complexity for Hadamard transform

We compute the total number of operations for the two cases.

	$\mathbf{R}H(2^r)$	$\mathbf{R}M_{2^r}^{(1)} \cdots M_{2^r}^{(r)}$
Multipl./column	2^r	2
Addition/column	$2^r - 1$	1
Total	$2^r(2^r + 2^r - 1) \approx 2^{2r+1}$	$3r2^r$

The complexity ratio is thus,

$$\frac{3r2^r}{2^r(2^r + 2^r - 1)} = \frac{3r}{2^{r+1} - 1}$$

For the RM(1,5) code (Mariner) the decoding requires $3r2^r = 480$ operations; standard array need the storage for $2^{32}/2^6 = 2^{26}$ coset leaders.

RM codes through Boolean functions

- Boolean functions map n binary inputs to a single binary output.
- More formaly $f : \mathbb{F}_2^n \to \mathbb{F}_2$ maps ($\mathbb{F}_2^n = GF(2)^n$)

$$(x_1, \ldots, x_n) \in \mathbb{F}_2^n \mapsto f(x) \in \mathbb{F}_2$$

- $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be represented as a polynomial in the ring

$$\mathbb{F}_2[x_1, \ldots, x_n]/ < x_1^2 = x_1, \ldots, x_n^2 = x_n >$$

- This ring is simply a set of all polynomials with binary coefficients in n indeterminates with property that $x_i^2 = x_i$.

Truth table -Example

The *truth table* of f is the evaluation of f for all possible inputs.

Example

E.g. for $f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3$

x_3	x_2	x_1	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Important $\mathbf{1}, x_1, x_2, x_3$ spans the RM(1,3) code !!

Boolean functions-definitions

- This may be formalized further by defining,

$$f(x) = \sum_{c \in \mathbb{F}_2^n} a_c x^c = \sum_{c \in \mathbb{F}_2^n} a_c x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n}, \quad c = (c_1, \ldots, c_n)$$

- Thus f is specified by the coefficients a_c
- There are 2^n different terms $x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n}$ for different c's. As a_c is binary it gives 2^{2^n} different functions in n variables.

Example

For $n = 3$ there are $2^8 = 256$ distinct functions specified by a_c,

$B_3 = \{a_0 1 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_4 x_1 x_2 \oplus a_5 x_1 x_3 \oplus a_6 x_2 x_3 \oplus a_7 x_1 x_2 x_3\}$

Higher order Reed-Muller codes

All the codewords of RM$(1, r)$ are of weight 2^{r-1}, apart from $\mathbf{0}$ and $\mathbf{1}$. Affine Boolean functions in r variables

- Generalization - t-th order Reed-Muller code

$$\mathbf{1}, \underbrace{x_1, \ldots, x_r}_{\text{linear terms}}, \underbrace{x_1 x_2, \ldots, x_{r-1} x_r}_{\text{quadratic terms}}, \ldots, \underbrace{x_1 \cdots x_t, \ldots, x_{r-t+1} \cdots x_r}_{\text{degree } t}$$

- The dimension of the basis is,

$$k = 1 + \binom{r}{1} + \binom{r}{2} + \cdots + \binom{r}{t}$$

All the vectors are linearly independent.

Higher order Reed-Muller codes- example

We consider RM(2,3) code.

x_3	x_2	x_1	$\mathbf{1}$	x_1x_2	x_1x_3	x_2x_3
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	1	0	0
1	0	0	1	0	0	0
1	0	1	1	0	1	0
1	1	0	1	0	0	1
1	1	1	1	1	1	1

Seven basis vectors - (8,7,2) code (recall uniqueness). 128 codewords out of 256 binary vectors of length 8 !

Constructing higher order Reed-Muller codes

Given an RM(t,r) code how do we construct an RM$(t+1,r+1)$?

$$RM(t+1,r+1)=\{(u,u+v):u\in RM(t+1,r), v\in RM(t,r)\}$$

In terms of generating matrices this is equivalent to:

$$G(t+1,r+1)=\begin{bmatrix} G(t+1,r) & G(t+1,r) \\ 0 & G(t,r) \end{bmatrix}$$

To prove this we need an easy result on Boolean functions,

$$f(x_1,\ldots,x_r,x_{r+1})=g(x_1,\ldots,x_r)+x_{r+1}h(x_1,\ldots,x_r)$$

for some g,h (decomposition).

Constructing higher order Reed-Muller codes II

E.g.,

$$\begin{aligned} f(x_1,x_2,x_3) &= x_1+x_2+x_1x_3+x_1x_2+x_1x_2x_3 \\ &= \underbrace{x_1+x_2+x_1x_2}_{g(x_1,x_2)} + x_3\underbrace{(x_1+x_1x_2)}_{h(x_1,x_2)} \end{aligned}$$

x_3	x_2	x_1	$\mathbf{1}$	x_1x_2	x_1x_3	x_2x_3	$g(x)$	$h(x)$	$f(x)$
0	0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	1	0	1	0	0	0	1	0	1
0	1	1	1	1	0	0	1	0	1
1	0	0	1	0	0	0			0
1	0	1	1	0	1	0			1
1	1	0	1	0	0	1			0
1	1	1	1	1	1	1			0

Proving the result on higher order RM

$$\boxed{RM(t+1,r+1) = \{(u,u+v) : u \in RM(t+1,r), v \in RM(t,r)\}}$$

- Codeword from $RM(t+1,r+1)$- evaluation of polynomial $f(x_1,\ldots,x_{r+1})$ of degree $\leqslant t+1$
- Decomp. $f(x_1,\ldots,x_{r+1}) = g(x_1,\ldots,x_r) + x_{r+1}h(x_1,\ldots,x_r)$ implies $\deg(g) \leqslant t+1$ and $\deg(h) \leqslant t$
- Need the association between functions and vectors !

Proving the result on higher order RM II

- $\mathbf{g}$, $\mathbf{h}$ vectors of length $2^r \leftrightarrow g(x_1, \ldots, x_r)$, $h(x_1, \ldots, x_r)$

$$\begin{array}{ccc} g(x_1, \ldots, x_r) & x_{r+1} h(x_1, \ldots, x_r) & \in \mathbb{F}_2[x_1, \ldots, x_{r+1}] \\ \updownarrow & \updownarrow & \updownarrow \\ (\mathbf{g}, \mathbf{g}) & (\mathbf{0}, \mathbf{h}) & \text{vectors of length } 2^{r+1} \end{array}$$

Thus $\mathbf{f} = (\mathbf{g}, \mathbf{g}) + (\mathbf{0}, \mathbf{h}) = (\mathbf{g}, \mathbf{g} + \mathbf{h})$

Important results on higher order RM codes

$\boxed{\text{What about the minimum distance of } RM(t, r),\ t > 1\ ?}$

Recall: For $t = 1$ we had $d(C) = 2^{r-1}$! Generalization ?

Theorem $RM(t, r)$ has minimum distance 2^{r-t}.

Proof Fix t and use induction on r. What is a typical codeword of $RM(t, r + 1)$? Details of the proof - exercise.

Another important result is given by,

Theorem The dual code of $RM(t, r)$ is $RM(r - t - 1, r)$.

Dual code proof (optional)

Proof Take $\mathbf{a} \in RM(r-t-1, r)$ and $\mathbf{b} \in RM(t, r)$.

Alternatively we may consider $a(x_1, \ldots, x_r)$ with $\deg(a) \leqslant r-t-1$ and $b(x_1, \ldots, x_r)$ with $\deg(b) \leqslant t$. Then $\deg(ab) \leqslant r-1$.

Thus, $\mathbf{ab} \in RM(r-1, r)$ and has even weight, $\mathbf{ab} \equiv 0 \bmod 2$!

Therefore, $RM(r-t-1, r) \subset RM(t, r)^{\perp}$. But,

$$\begin{aligned} & \dim RM(r-t-1, r) + \dim RM(t, r) \\ = \; & 1 + \binom{r}{1} + \cdots + \binom{r}{r-t-1} + 1 + \binom{r}{1} + \cdots + \binom{r}{t} = 2^r \end{aligned}$$

So $RM(r-t-1, r) = RM(t, r)^{\perp}$ ($\dim RM(t, r)^{\perp} = 2^r - \dim RM(t, r)$)

Self-Dual codes - Motivation

- Very interesting class from theoretical and practical point of view.
  The main properties:
  - In some cases easily decodable
  - Additional algebraic structure
  - Problem of classification of these codes and finding the minimum weight for a given length

  No constructive method for finding these codes, different constructions for each n, e.g. the Golay code.

Self-orthogonal codes

Definition

A linear code C is self-orthogonal if $C \subset C^{\perp}$.

Each codeword orthogonal to every other codeword !

Example

The matrix,

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

generates a self-orthogonal code C. One can check that $\mathbf{c}_i \mathbf{c_j} = 0$.

- But more importantly $GG^T = \mathbf{0}$!! Coincidence ?

Fast Hadamard transform RM codes and Boolean functions Self-Dual codes

Self-orthogonal codes

Theorem

(Lemma 4.5) A linear code C is self-orthogonal iff $GG^T = \mathbf{0}$

Proof.

(sketch) Assume $C \subset C^{\perp}$. Let $\mathbf{r}_i$ be a row of G. Then,

$$\mathbf{r}_i \in C; \quad C \subset C^{\perp} \Rightarrow \mathbf{r}_i \in C^{\perp}$$

Since G is a parity check of $C^{\perp}$ then $G\mathbf{r}_i^T = 0$. As this is true for any $\mathbf{r}_i$ so $GG^T = 0$. □

Self-dual codes

Definition

A linear code C is self-dual if $C = C^{\perp}$

A self-dual code is clearly self-orthogonal but the converse need not be true.

E.g. (5,2) code in the previous example cannot be self-dual. Why ?

Self-dual codes - example

The generator matrix,

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

defines a binary self-dual $(8, 4)$ code C.

For binary codes needs to check that $GG^T = 0$ and $n = 2k$

Lemma If $G = [I_k B]$ for a self-dual (n, k) code C then $BB^T = -I_k$